

EC

NUSC Technical Report 5341

NUSC Technical Report 5341



12

Numerical Solutions of Initial Value Problems

ADA 028497

Ding Lee
Systems Analysis Department



23 July 1978

NUSC

NAVAL UNDERWATER SYSTEMS CENTER
Newport, Rhode Island • New London, Connecticut

Approved for public release; distribution unlimited.

PREFACE

This work was supported by NUSC as a continuation of long-term educational support for the author's dissertation (NUSC TR 4775, May 1974). The author wishes to thank Professor Stanley Preiser, Philip Hsiang, and Robert Casal of the Polytechnic Institute of New York, Brooklyn, New York and Mr. J. L. Malone of Westinghouse Electric Corporation, Annapolis, Maryland. These men evaluated the author's technique on the IBM 360/370 and the CDC 6600 and validated his computations which had been performed on the Univac 1108.

The Technical Reviewer for this report was Dr. Sung-Hwan Ko, Code 316.

REVIEWED AND APPROVED: 23 July 1976


W. L. Clearwaters
Associate Technical Director
for Plans and Analysis

The author of this report is located at the New London Laboratory, Naval Underwater Systems Center, New London, Connecticut 06320.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM												
(14) MUSC- TR-5341	1. REPORT NUMBER	2. GOVT ACCESSION NO.												
6. TITLE (and Subtitle)		3. RECIPIENT'S CATALOG NUMBER												
6. NUMERICAL SOLUTIONS OF INITIAL VALUE PROBLEMS		5. TYPE OF REPORT & PERIOD COVERED <i>(9) Technical Report</i>												
(15) Ding/Lee	7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s)												
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Underwater Systems Center New London Laboratory New London, CT 06320		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS												
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE <i>(11) 23 Jul 76</i>												
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		13. SECURITY CLASS. (of this report) UNCLASSIFIED												
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE												
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.														
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) <i>D D C R-PAPLID AUG 19 1976 R-LGCLTLD C</i>														
18. SUPPLEMENTARY NOTES														
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">Numerical Methods</td> <td style="width: 33%;">Nonlinear Multistep Methods</td> <td style="width: 33%;">Step Size</td> </tr> <tr> <td>Initial Value Problems</td> <td>Linear Multistep Methods</td> <td>Computer Solution of</td> </tr> <tr> <td>Differential Equations</td> <td>FORTRAN Program for NLMS</td> <td>Differential</td> </tr> <tr> <td>Stiff Equations</td> <td></td> <td>Equations</td> </tr> </table>			Numerical Methods	Nonlinear Multistep Methods	Step Size	Initial Value Problems	Linear Multistep Methods	Computer Solution of	Differential Equations	FORTRAN Program for NLMS	Differential	Stiff Equations		Equations
Numerical Methods	Nonlinear Multistep Methods	Step Size												
Initial Value Problems	Linear Multistep Methods	Computer Solution of												
Differential Equations	FORTRAN Program for NLMS	Differential												
Stiff Equations		Equations												
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) <p>A family of nonlinear multistep (NLMS) numerical methods has been developed that is particularly effective for solving stiff ordinary differential equations. These methods offer the advantages of avoiding small step sizes and being A-stable in the Dahlquist sense. These advantages over existing conventional and stiff methods have been demonstrated by the present author with a number of test cases. These same → next page</p>														

20. (Cont'd)

~~cont.~~ methods can also be used to solve nonasymptotically stable differential equations since they are a generalization of Linear Multistep (LMS) methods.

This report presents a detailed formulation of NLMS methods and discusses a FORTRAN V computer package specifically developed to implement NLMS methods. The package, also available in ANSI FORTRAN, is presently operational on Univac 1108, IBM 360/370, and CDC 6600 computers. Several desirable features are included in the computer program, namely, a fixed or variable step size, self-start, a selection of characteristic polynominal coefficients, predict-and-correct m times, and the inclusion of LMS methods. Whenever matrix A is a function of t, a periodic decomposition technique is employed. Otherwise, a decomposition technique can be selected by the user. Nonlinear multistep methods and their associated features constitute a powerful means for solving initial value problems of stiff, nonstiff, linear, and nonlinear ordinary differential equations. The validity of NLMS methods has been proved theoretically while the effectiveness of the computer program will be supported by the numerical evidence to be presented.

TABLE OF CONTENTS

	Page
1. INTRODUCTION	1
2. PRELIMINARY CONSIDERATIONS	3
3. FORMULATION	5
4. STEP SIZES OF NLMS METHODS	13
5. FEATURES	17
5.1 Variable Step Size and Fixed Step Size	17
5.2 The PC ^m Procedure	17
5.3 Self-Starting	18
5.4 Selection of Characteristic Coefficients α_i	18
5.5 A as a Function of t	21
5.6 Inclusion of LMS Methods	21
5.7 Miscellaneous Features	22
6. PROGRAMMING ASPECTS	23
6.1 Notes to Users	23
6.2 Formats of User-Supplied Subroutines	25
7. USER'S GUIDE	33
8. NUMERICAL RESULTS AND INPUT SETUPS	35
8.1 Problem Identification	35
8.2 Selection of Test Problems	35
8.3 Problems, Input Setups, and Solutions	36
9. CONCLUSIONS	57
10. REFERENCES	59
APPENDIX — NLMS COMPUTER PROGRAMS	A-1

NUMERICAL SOLUTIONS OF INITIAL VALUE PROBLEMS

1. INTRODUCTION

To solve nonstiff differential equations, the conventional Runge-Kutta and linear multistep methods are typically employed. These conventional methods are impractical for solving stiff equations because prohibitively small step sizes are required for accuracy. To overcome this difficulty, nonlinear multistep (NLMS) methods can be applied; NLMS methods have been shown to have advantages over existing stiff methods. It is desirable to be prepared with a variety of effective methods for handling both stiff and nonstiff equations. The search for such an efficient program will probably never end. To be effective, a package must be reliable, easy to modify, and convenient to use. In addition, other features are desirable. To satisfy these needs, the NLMS method was developed. It is now known that strongly stable NLMS methods are consistent and, therefore, convergent. The complete theory of NLMS methods has been published.¹

This report will first outline some preliminary considerations and assumptions and then describe the formulation of NLMS methods along with the various built-in features. Since the principal objective was to solve stiff equations, variable-order techniques were not utilized because NLMS methods of all orders were considered to be effective.

The various features are described and illustrated by problems. A section of numerical results describes the selection of these problems from well-known sources, categorizes them in various classes, and then summarizes their characteristics in tabular form. Various NLMS methods are applied to solve these problems with given initial values. The computed solutions are next compared with exact solutions. The computational accuracy is measured mainly by relative error, which will be defined in the next section. An error of 10^{-10} is used as an upper bound to acceptable performance in test examples. Numerical results are printed out to eight significant digits. The User's Guide is intended to help the user become familiar with the inputs. To assist in this objective, the way to start a problem with various sample setups is described within each test problem. Only FORTRAN V inputs are described since ANSI FORTRAN inputs are made optional to the user. The usage of a few user-supplied or optional subroutines is also described within each test problem, which exercises various

TR 5341

options. Programs were originally written in FORTRAN V language and checked out on the Univac 1108 computer using double-precision arithmetic by means of the EXEC 8 operating system.

These programs are also available on IBM 360/370 and CDC 6600 in FORTRAN. A version of the program is made available in ANSI FORTRAN, where the external and the adjustable dimensions were purposely eliminated. Detailed operational descriptions of the IBM and CDC machines are not included. In the appendix, a program listing of both FORTRAN V and ANSI FORTRAN are given.

2. PRELIMINARY CONSIDERATIONS

In this section, we define the problems under consideration, give the norm used for comparison, and state some assumptions.

1. Let us consider the initial value problem of a system of first-order ordinary differential equations of the form

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}); \quad \mathbf{y}(t_0) = \mathbf{y}_0. \quad (2.1)$$

which can also be written as

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{g}(t, \mathbf{y}); \quad \mathbf{y}(t_0) = \mathbf{y}_0 \quad (2.2)$$

in the region R , defined by $-\infty < a \leq t \leq b < \infty$; $\|\mathbf{y}\| < \infty$. The matrix \mathbf{A} is nonsingular, either a constant or a function of t . We assume that our initial value problems satisfy the conditions required by the existence and uniqueness theorem; therefore, \mathbf{f} and \mathbf{g} both satisfy a Lipschitz condition with Lipschitz constants L^* and L , respectively, and \mathbf{f} is continuous in R .

2. Subject to the consideration 1., we consider only first-order equations. This is not a restriction because higher order equations can always be decomposed into a system of first-order equations. Equations that are stiff, nonstiff, linear, nonlinear, homogeneous, and nonhomogeneous are all taken into consideration. We regard every problem as a system of equations.

3. Let \mathbf{y}_C be a computed solution vector and \mathbf{y}_E be an exact solution vector. The norm used here to measure the error is defined as follows:

$$\|\mathbf{y}\|_{\infty} = \lim_{p \rightarrow \infty} \|\mathbf{y}\|_p = \max_j |y_j|. \quad (2.3)$$

We define the relative error E to mean

$$E = \max_i \left\{ \frac{\frac{(y_i)_c - (y_i)_E}{(y_i)_E}}{\frac{(y_i)_c}{(y_i)_E}} \right\}. \quad (2.4)$$

TR 5341

In the event that an absolute error is considered, it possesses the following definition:

$$E = \max_i \left\{ (y_i)_c - (y_i)_E \right\} . \quad (2.5)$$

3. FORMULATION

The linear multistep methods of step K can be expressed by²

$$\sum_{i=0}^K \alpha_i y_{n+i} = h \sum_{i=0}^K \beta_i f_{n+i}, \quad (3.1)$$

where $\alpha_K \neq 0$, $|\alpha_0| + |\beta_0| > 0$, and α_i and β_i are constants independent of mesh size h . An LMS method can solve equation (2.1) effectively when $\|\frac{\partial f}{\partial y}\|$ is small.

A generalization of equation (3.1) leads to the NLMS methods of step K in the form

$$\sum_{i=0}^K \alpha_i e^{\mathbf{A}h(K-i)} y_{n+i} = h \sum_{i=0}^K \phi_{Ki}(\mathbf{A}h) g_{n+i}, \quad (3.2)$$

where $\alpha_K \neq 0$, $|\alpha_0| + |\lambda(\phi_{K0}(\mathbf{A}h))| > 0$, and α_i are independent of h , but $\phi_{Ki}(\mathbf{A}h)$ are functions of h and nonsingular \mathbf{A} . (The Generalized Adams-Basforth (GAB) and the Generalized Adams-Moulton (GAM) methods for $K = 1, 2$ are suggested by the work of Certaine.³)

The NLMS methods are designed to solve equation (2) effectively when $\|\frac{\partial f}{\partial y}\| = \|\mathbf{A} + \frac{\partial g}{\partial y}\|$ is large. For \mathbf{A} , which is a constant matrix, we write equation (2.2) as

$$\frac{d}{dt} (e^{-\mathbf{A}t} y) = e^{-\mathbf{A}t} g(t, y); \quad y(t_0) = y_0. \quad (3.3)$$

Integration of equation (3.3) over the interval $[t_n, t_{n+1}]$ gives

$$y(t_{n+1}) = e^{i\mathbf{A}h} y(t_n) + \int_{t_n}^{t_{n+1}} e^{\mathbf{A}(t_{n+1}-t')} g(t', y) dt'. \quad (3.4)$$

Expressing $\mathbf{g}(t', \mathbf{y})$ in a Taylor series expansion around t_n , and, next, substituting it into equation (3.4), we obtain

$$\mathbf{y}(t_{n+i}) = e^{i\mathbf{A}h} \mathbf{y}(t_n) + \sum_{j=0}^{\infty} \frac{\Xi_i^j(\mathbf{A}h)}{j!} \mathbf{g}^{(j)}(t_n, \mathbf{y}(t_n)), \quad (3.5)$$

where

$$\Xi_i^j(\mathbf{A}h) = \int_{t_n}^{t_{n+i}} e^{\mathbf{A}(t_{n+i} - t')} (t' - t_n)^j dt. \quad (3.6)$$

We define the nonlinear multistep operator $\mathcal{L}_N[\mathbf{y}(t); h]$ to be

$$\begin{aligned} \mathcal{L}_N[\mathbf{y}(t); h] &= \sum_{i=0}^K \alpha_i e^{\mathbf{A}h(K-i)} \mathbf{y}(t + ih) \\ &\quad - h \sum_{i=0}^K \phi_{Ki}(\mathbf{A}h) \mathbf{g}(t + ih, \mathbf{y}). \end{aligned} \quad (3.7)$$

Expanding $\mathbf{g}(t + ih, \mathbf{y})$ in powers of h at $t = t_n$ yields

$$\mathbf{g}(t + ih, \mathbf{y}) = \sum_{j=0}^{\infty} \frac{(ih)^j}{j!} \mathbf{g}^{(j)}(t_n, \mathbf{y}(t_n)). \quad (3.8)$$

If we substitute $\mathbf{g}(t + ih, \mathbf{y})$ into equation (3.7) and use equation (3.5) for $\mathbf{y}(t + ih)$ in (3.7), we obtain

$$\begin{aligned}
 \mathcal{L}_N[y(t); h] &= \sum_{i=0}^K \alpha_i e^{\mathbf{A}h(K-i)} \left[e^{i\mathbf{A}h} y + \sum_{j=0}^{\infty} \frac{\mathbf{I}_i^j(\mathbf{A}h)}{j!} \mathbf{g}^{(j)} \right] \\
 &\quad - h \sum_{i=0}^K \phi_{Ki}(\mathbf{A}h) \left[\sum_{j=0}^{\infty} \frac{(ih)^j}{j!} \mathbf{g}^{(j)} \right] \\
 &= \left\{ \sum_{i=0}^K \alpha_i e^{\mathbf{A}h(K-i)} e^{i\mathbf{A}h} y \right\} + \sum_{j=0}^{\infty} \mathbf{C}_j(\mathbf{A}h) \mathbf{g}^{(j)},
 \end{aligned}$$

where

$$\mathbf{C}_j(\mathbf{A}h) = \sum_{i=0}^K \alpha_i e^{\mathbf{A}h(K-i)} \left[\frac{\mathbf{I}_i^j(\mathbf{A}h)}{j!} \right] - h \sum_{i=0}^K \frac{(ih)^j}{j!} \phi_{Ki}(\mathbf{A}h).$$

A consistent NLMS method is said to be of order p if

$$\mathbf{C}_0 = \mathbf{C}_1 = \dots = \mathbf{C}_p = \mathbf{0}, \text{ but } \mathbf{C}_{p+1} \neq \mathbf{0}.$$

By mathematical induction, it is seen that

$$\frac{\mathbf{A}^{j+1} \mathbf{I}_i^j(\mathbf{A}h)}{j!} = e^{i\mathbf{A}h} - \sum_{l=0}^j \frac{(i\mathbf{A}h)^l}{l!}.$$

Therefore,

$$\begin{aligned}
 -\mathbf{A}^{j+1} \mathbf{C}_j(\mathbf{A}h) &= \sum_{i=0}^K \alpha_i e^{\mathbf{A}h(K-i)} \sum_{l=0}^j \frac{(i\mathbf{A}h)^l}{l!} + \\
 &\quad \frac{(\mathbf{A}h)^{j+1}}{j!} \cdot \sum_{i=0}^K (i)^j \phi_{Ki}(\mathbf{A}h) = \mathbf{0}. \tag{3.9}
 \end{aligned}$$

A method of equation (3.2) is said to be consistent³ if

$$\max_n \left\| \sum_{i=0}^K \alpha_i e^{\mathbf{A}h(K-i)} \mathbf{y}_{n+i} - h \sum_{i=0}^K \phi_{Ki}(\mathbf{A}h) \mathbf{g}_{n+i} \right\|$$

is small as $h \rightarrow 0$.

The requirement that $\mathbf{C}_j(\mathbf{A}h) = \mathbf{0}$ for $j = 0, 1, \dots, p$ yields the consistency and permits formulating the NLMS method in the following matrix form:

$$\mathbf{E}\psi = -\mathbf{H}\mathbf{K}\phi, \quad (3.10)$$

where \mathbf{E} , ψ , \mathbf{H} , \mathbf{K} , and ϕ are described by the expanded forms of both explicit and implicit forms.

In expanded forms, the explicit schemes satisfy

$$\begin{pmatrix} I & I & \dots & I \\ I & I + \mathbf{A}h & \dots & I + K\mathbf{A}h \\ \vdots & \vdots & & \vdots \\ I & \sum_{m=0}^{p-1} \frac{(\mathbf{A}h)^m}{m!} & \dots & \sum_{m=0}^{p-1} \frac{(K\mathbf{A}h)^m}{m!} \end{pmatrix} \begin{pmatrix} \alpha_0 e^{K\mathbf{A}h} \\ \alpha_1 e^{(K-1)\mathbf{A}h} \\ \vdots \\ \alpha_K I \end{pmatrix}$$

$$= - \begin{pmatrix} \frac{\mathbf{A}h}{0!} & & & \\ & \frac{(\mathbf{A}h)^2}{1!} & & \\ & & \ddots & \\ & & & \frac{(\mathbf{A}h)^p}{(p-1)!} \end{pmatrix} \begin{pmatrix} I & I & \dots & I \\ 0 & I & \dots & (K-1)I \\ \vdots & \vdots & & \vdots \\ 0 & I & \dots & (K-1)^{p-1} I \end{pmatrix} \begin{pmatrix} \phi_{K0} \\ \phi_{K1} \\ \vdots \\ \phi_{K,K-1} \end{pmatrix} \quad (3.11)$$

and the implicit schemes satisfy

$$\begin{pmatrix}
 I & I & \dots & I \\
 I & I + \mathbf{A}h & \dots & I + K\mathbf{A}h \\
 \vdots & \vdots & & \vdots \\
 I & \sum_{m=0}^p \frac{(\mathbf{A}h)^m}{m!} & \dots & \sum_{m=0}^p \frac{(K\mathbf{A}h)^m}{m!}
 \end{pmatrix}
 \begin{pmatrix}
 \alpha_0 e^{K\mathbf{A}h} \\
 \alpha_1 e^{(K-1)\mathbf{A}h} \\
 \vdots \\
 \alpha_K I
 \end{pmatrix}
 = - \begin{pmatrix}
 \frac{\mathbf{A}h}{0!} & & & \\
 & \frac{(\mathbf{A}h)^2}{1!} & & \\
 & & \ddots & \\
 & & & \frac{(\mathbf{A}h)^{p+1}}{p!}
 \end{pmatrix}
 \begin{pmatrix}
 I & I & \dots & I \\
 0 & I & \dots & KI \\
 \vdots & \vdots & & \vdots \\
 0 & I & \dots & K^p I
 \end{pmatrix}
 \begin{pmatrix}
 \phi_{K0} \\
 \phi_{K1} \\
 \vdots \\
 \phi_{KK}
 \end{pmatrix}. \quad (3.12)$$

We determine $\phi_{Ki}(\mathbf{A}h)$ without loss of generality by selecting $\alpha_K = 1$

and then requiring that the condition of strong stability be realized in selecting α_i . The $\phi_{Ki}(\mathbf{A}h)$ are determined using the above matrix formula, which can be considered as a matrix equation for the K -step, p -th-order method ($K \geq 1$, $p \geq 0$). Below we show that $\phi_{Ki}(\mathbf{A}h)$ can be determined by, first, listing the formulas for the remaining $\phi_{Ki}(\mathbf{A}h)$, and second, exhibiting an explicit NLMS method of order 2. From formula (3.10) for nonsingular \mathbf{H} and \mathbf{K} , we get

$$\phi = -\mathbf{K}^{-1} \mathbf{H}^{-1} \mathbf{E} \psi, \quad (3.13)$$

where ϕ is a vector of dimension $(K-1)$ or K , depending upon whether the scheme is explicit or implicit. Explicit schemes are the following:

$$K = p = 1, \phi_{1,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-1} (\alpha_0 e^{\mathbf{A}h} + I) \quad (3.14)$$

$$K = p = 2, \quad \phi_{2,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-2} \left[\alpha_0 (\mathbf{A}h - I) e^{2\mathbf{A}h} - \alpha_1 e^{\mathbf{A}h} - (I + \mathbf{A}h) \right] \quad (3.15)$$

$$\phi_{2,1}(\mathbf{A}h) = -(\mathbf{A}h)^{-2} \left[\alpha_0 e^{2\mathbf{A}h} + \alpha_1 (I + \mathbf{A}h) e^{\mathbf{A}h} + (I + 2\mathbf{A}h) \right] \quad (3.16)$$

$$K = p = 3, \quad \phi_{3,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-3} \left[\alpha_0 \left(I - \frac{3}{2}\mathbf{A}h + (\mathbf{A}h)^2 \right) e^{3\mathbf{A}h} \right.$$

$$\left. + \alpha_1 \left(I - \frac{\mathbf{A}h}{2} \right) e^{2\mathbf{A}h} + \alpha_2 \left(I + \frac{\mathbf{A}h}{2} \right) e^{\mathbf{A}h} + I + \frac{3}{2}\mathbf{A}h + (\mathbf{A}h)^2 \right] \quad (3.17)$$

$$\begin{aligned} \phi_{3,1}(\mathbf{A}h) = & -(\mathbf{A}h)^{-3} \left[-2\alpha_0 (I - \mathbf{A}h) e^{3\mathbf{A}h} + \alpha_1 (-2I + (\mathbf{A}h)^2) e^{2\mathbf{A}h} \right. \\ & \left. - 2\alpha_2 (I + \mathbf{A}h) e^{\mathbf{A}h} - (2I + 4\mathbf{A}h + 3(\mathbf{A}h)^2) \right], \end{aligned} \quad (3.18)$$

$$\begin{aligned} \phi_{3,2}(\mathbf{A}h) = & -(\mathbf{A}h)^{-3} \left[\alpha_0 \left(I - \frac{\mathbf{A}h}{2} \right) e^{3\mathbf{A}h} + \alpha_1 \left(I + \frac{\mathbf{A}h}{2} \right) e^{2\mathbf{A}h} \right. \\ & \left. + \alpha_2 \left(I + \frac{3}{2}\mathbf{A}h + (\mathbf{A}h)^2 \right) e^{\mathbf{A}h} + (I + \frac{5}{2}\mathbf{A}h + 3(\mathbf{A}h)^2) \right]. \end{aligned} \quad (3.19)$$

Implicit schemes are the following:

$$K = p = 1,$$

$$\phi_{1,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-2} \left[\alpha_0 (\mathbf{A}h - I) e^{\mathbf{A}h} - I \right], \quad (3.20)$$

$$\phi_{1,1}(\mathbf{A}h) = -(\mathbf{A}h)^{-2} \left[\alpha_0 e^{\mathbf{A}h} + (I + \mathbf{A}h) \right]. \quad (3.21)$$

$$K = p = 2,$$

$$\begin{aligned} \phi_{2,0}(\mathbf{A}h) = & -(\mathbf{A}h)^{-3} \left[\alpha_0 \left(I - \frac{3}{2}\mathbf{A}h + (\mathbf{A}h)^2 \right) e^{2\mathbf{A}h} \right. \\ & \left. + \alpha_1 \left(I - \frac{\mathbf{A}h}{2} \right) e^{\mathbf{A}h} + (I + \frac{\mathbf{A}h}{2}) \right], \end{aligned} \quad (3.22)$$

$$\begin{aligned} \phi_{2,1}(\mathbf{A}h) = & -(\mathbf{A}h)^{-3} \left[\alpha_0 (-2I + 2\mathbf{A}h) e^{2\mathbf{A}h} \right. \\ & \left. + \alpha_1 (-2I + (\mathbf{A}h)^2) e^{\mathbf{A}h} - 2(I + \mathbf{A}h) \right], \end{aligned} \quad (3.23)$$

$$\begin{aligned}\phi_{2,2}(\mathbf{A}h) = & -(\mathbf{A}h)^{-3} \left[\alpha_0 \left(I - \frac{\mathbf{A}h}{2} \right) e^{2\mathbf{A}h} + \alpha_1 \left(I + \frac{\mathbf{A}h}{2} \right) e^{\mathbf{A}h} \right. \\ & \left. + \left(I + \frac{3}{2}\mathbf{A}h + (\mathbf{A}h)^2 \right) \right].\end{aligned}\quad (3.24)$$

$K = p = 3$,

$$\begin{aligned}\phi_{3,0}(\mathbf{A}h) = & -(\mathbf{A}h)^{-4} \left[\alpha_0 \left(-I + 2\mathbf{A}h - \frac{11}{6}(\mathbf{A}h)^2 + (\mathbf{A}h)^3 \right) e^{3\mathbf{A}h} \right. \\ & + \alpha_1 \left(-I + \mathbf{A}h - \frac{1}{3}(\mathbf{A}h)^2 \right) e^{2\mathbf{A}h} + \alpha_2 \left(-I + \frac{1}{6}(\mathbf{A}h)^2 \right) e^{\mathbf{A}h} \\ & \left. + \left(-I - \mathbf{A}h - \frac{1}{3}(\mathbf{A}h)^2 \right) \right]\end{aligned}\quad (3.25)$$

$$\begin{aligned}\phi_{3,1}(\mathbf{A}h) = & -(\mathbf{A}h)^{-4} \left[\alpha_0 \left(3I - 5\mathbf{A}h + 3(\mathbf{A}h)^2 \right) e^{3\mathbf{A}h} \right. \\ & + \alpha_1 \left(3I - 2\mathbf{A}h - \frac{1}{2}(\mathbf{A}h)^2 + (\mathbf{A}h)^3 \right) e^{2\mathbf{A}h} \\ & + \alpha_2 \left(3I + \mathbf{A}h - (\mathbf{A}h)^2 \right) e^{\mathbf{A}h} \\ & \left. + \left(3I + 4\mathbf{A}h + \frac{3}{2}(\mathbf{A}h)^2 \right) \right]\end{aligned}\quad (3.26)$$

$$\begin{aligned}\phi_{3,2}(\mathbf{A}h) = & -(\mathbf{A}h)^{-4} \left[\alpha_0 \left(-3I + 4\mathbf{A}h - \frac{3}{2}(\mathbf{A}h)^2 \right) e^{3\mathbf{A}h} \right. \\ & + \alpha_1 \left(-3I + \mathbf{A}h + (\mathbf{A}h)^2 \right) e^{2\mathbf{A}h} \\ & + \alpha_2 \left(-3I - 2\mathbf{A}h + \frac{1}{2}(\mathbf{A}h)^2 + (\mathbf{A}h)^3 \right) e^{\mathbf{A}h} \\ & \left. + \left(-3I - 5\mathbf{A}h - 3(\mathbf{A}h)^2 \right) \right]\end{aligned}\quad (3.27)$$

$$\begin{aligned}\phi_{3,3}(\mathbf{A}h) = & -(\mathbf{A}h)^{-4} \left[\alpha_0 \left(I - \mathbf{A}h + \frac{1}{3}(\mathbf{A}h)^2 \right) e^{3\mathbf{A}h} + \alpha_1 \left(I - \frac{1}{6}(\mathbf{A}h)^2 \right) e^{2\mathbf{A}h} \right. \\ & \left. + \alpha_2 \left(I + \mathbf{A}h + \frac{1}{3}(\mathbf{A}h)^2 \right) e^{\mathbf{A}h} + \left(I + 2\mathbf{A}h + \frac{11}{6}(\mathbf{A}h)^2 + (\mathbf{A}h)^3 \right) \right]\end{aligned}\quad (3.28)$$

In the explicit case when $K = p = 2$, it is seen that

$$\mathbf{H} = \begin{pmatrix} \mathbf{A}_h & \mathbf{0} \\ \mathbf{0} & (\mathbf{A}_h)^2 \end{pmatrix}, \quad \mathbf{H}^{-1} = (\mathbf{A}_h)^{-2} \begin{pmatrix} \mathbf{A}_h & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix};$$

$$\mathbf{K} = \begin{pmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \quad \mathbf{K}^{-1} = \begin{pmatrix} \mathbf{I} & -\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{pmatrix};$$

$$\mathbf{E} = \begin{pmatrix} \mathbf{I} & & & \\ & \mathbf{I} & & \\ & & \mathbf{I} & \\ & & & \mathbf{I} + \mathbf{A}_h & \mathbf{I} + 2\mathbf{A}_h \end{pmatrix};$$

and, therefore,

$$\phi = \begin{pmatrix} \phi_{2,0} \\ \phi_{2,1} \end{pmatrix} = -(\mathbf{A}_h)^{-2} \begin{pmatrix} \alpha_0 (\mathbf{A}_h - \mathbf{I}) e^{2\mathbf{A}_h} - \alpha_1 e^{\mathbf{A}_h} - \alpha_2 (\mathbf{I} + \mathbf{A}_h) \\ \alpha_0 e^{2\mathbf{A}_h} + \alpha_1 (\mathbf{I} + \mathbf{A}_h) e^{\mathbf{A}_h} + \alpha_2 (\mathbf{I} + 2\mathbf{A}_h) \end{pmatrix} \quad (3.29)$$

The selection of $\alpha_2 = 1$, $\alpha_1 = -1$, and $\alpha_0 = 0$ leads to the GAB method which gives

$$\phi = \begin{pmatrix} \phi_{2,0} \\ \phi_{2,1} \end{pmatrix} = -(\mathbf{A}_h)^{-2} \begin{pmatrix} e^{\mathbf{A}_h} - (\mathbf{I} + \mathbf{A}_h) \\ -(\mathbf{I} + \mathbf{A}_h) e^{\mathbf{A}_h} + (\mathbf{I} + 2\mathbf{A}_h) \end{pmatrix} \quad (3.30)$$

However, the eigenvalues in stiff equations differ greatly in magnitude. As a consequence, double-precision arithmetic should be used in calculating ϕ .

4. STEP SIZES OF NLMS METHODS

Nonlinear multistep methods are designed to avoid the use of small stepsizes where $\mathbf{g}(t, \mathbf{y})$ is a slowly varying function that can be approximated by a low-order polynomial in t . To demonstrate this, we give an analysis below and show that a larger step size can be selected using NLMS methods than using LMS methods. From equation (3.2), since $\alpha_k \neq 0$, we can write

$$\mathbf{y}_{n+k} = \frac{1}{\alpha_K} \left\{ - \sum_{i=0}^{K-1} \alpha_i e^{\mathbf{A}h(K-i)} \mathbf{y}_{n+i} + h \sum_{i=0}^K \phi_{Ki}(\mathbf{A}h) \mathbf{g}_{n+i} \right\}, \quad (4.1)$$

which is of the form

$$\mathbf{y} = G(\mathbf{y}),$$

where $\mathbf{y} = \mathbf{y}_{n+k}$. The successive iterative form gives

$$\mathbf{y}^{(Y+1)} = G(\mathbf{y}^{(Y)}) \quad (4.2)$$

for any initial vector $\mathbf{y}^{(0)}$.

Let $G(\mathbf{y})$ be defined for $\|\mathbf{y}\| < \infty$, and let there exist a constant k such that $0 \leq k < 1$. Then, $G(\mathbf{y})$ satisfies the condition

$$\|G(\mathbf{y}^*) - G(\mathbf{y})\| < k \|\mathbf{y}^* - \mathbf{y}\|. \quad (4.3)$$

Using the definition of $G(\mathbf{y})$, formula (4.1), and the fact that $\mathbf{g}(t, \mathbf{y})$ satisfies the Lipschitz condition with Lipschitz constant L , we see that condition (4.3) is satisfied by

$$k = \frac{h \|\phi_{KK}(\mathbf{A}h)\|}{\alpha_K} L \quad (4.4)$$

for sufficiently small h and for all $\|\mathbf{A}\| < \infty$.

For the iterative procedure (4.2) to converge for arbitrary initial $\mathbf{y}^{(0)}$, k is required to be less than 1:

$$k < 1 \rightarrow \frac{h \parallel \Phi_{KK}(\mathbf{A}h) \parallel}{\alpha_K} L < 1. \quad (4.5)$$

Conventionally, when using LMS K-step methods with $\alpha_K = 1$, we select h such that

$$\beta_K h L^* \sim k \quad (< 1). \quad (4.6)$$

Similary, for NLMS K-step methods, we select h_N to satisfy condition (4.5):

$$\parallel \Phi_{KK}(\mathbf{A}h_N) \parallel h_N L \sim k. \quad (4.7)$$

Combining (4.6) and (4.7), we see that

$$h_N = \parallel \Phi_{KK}^{-1}(\mathbf{A}h_N) \parallel \beta_K \frac{L^*}{L} h. \quad (4.8)$$

For $\parallel \Phi_{KK}^{-1}(\mathbf{A}h_N) \parallel \beta_K$ not too small, we know that $L^* \gg L$; therefore, $h_N \gg h$. This tells us that we can choose a much larger step size using NLMS then we can choose using LMS.

The quantity h_N can be selected to satisfy

$$h_N < \frac{1}{\parallel \Phi_{KK}(\mathbf{A}h_N) \parallel \cdot \parallel \frac{\partial \mathbf{g}}{\partial \mathbf{y}} \parallel}. \quad (4.9)$$

In the event that $\parallel \frac{\partial \mathbf{g}}{\partial \mathbf{y}} \parallel = 0$, we can use a large h_N . This advantage is demonstrated by the problem below.

We use problem 1 from section 8.3 to demonstrate the step size choice. The problem is

$$\mathbf{y}' = -100\mathbf{y} + (1 + t^2); \mathbf{y}(0) = 1.$$

Table 4-1 compares two methods of solution.

Table 4-1. Comparison of Two Methods for Solving Problem 1

	Adams-Bashforth	Explicit NLMS	Remarks
$\left\ \frac{\partial \mathbf{F}}{\partial \mathbf{y}} \right\ $	100	0	$\mathbf{F} = \begin{cases} \mathbf{f}(t, \mathbf{y}) & \text{AB} \\ \mathbf{g}(t, \mathbf{y}) & \text{for NLMS} \end{cases}$
Step Number	3	3	
Step Size	2^{-8}	2.5	$h_N = 640h$
Tmax	10	10	
Solution Vector	.1008 0020 + 01	.1008 0020 + 01	
Exact Solution	.1008 0020 + 01		
Total Steps	2560	1	

5. FEATURES

5.1 VARIABLE STEP SIZE AND FIXED STEP SIZE

Variable step size is a desirable feature that is incorporated into this approach. By controlling the step size, the solution may be obtained quicker through the use of a larger step size. In some instances, the user may desire to examine the solution at a specified time value. The variable-step-size technique may not meet this requirement because a larger step can often bypass that point. This is commonly encountered in all variable-step-size programs. Although NLMS methods are designed to avoid the use of small step sizes, the program is designed with this option. However, the user can still use the variable step size if he desires and select the maximum step size HMAX.

To exercise this option, we require that the indicator IPC be defined as follows:

$$\begin{aligned} \text{IPC} = 0: \text{ fixed step size with INDEX} &= \begin{cases} 0 \text{ explicit} \\ 1 \text{ implicit} \end{cases} \\ \text{IPC} \neq 0: \text{ variable step size; } \text{PC}^m. \end{aligned}$$

If the present step size h needs a change, it is changed by an amount rh where $1/2 \leq r \leq 2$. To save computing time, we change h by halving or doubling, depending upon the need. This is handled automatically by the program.

An example to demonstrate this feature is given in problem 1 of section 8.3. Closely connected with the variable-step-size procedure is the PC^m procedure, which will be described in the next section.

5.2 THE PC^m PROCEDURE

The PC^m procedure stands for "predict and correct m times." This procedure has been widely used to achieve efficiency. Henrici² discussed a single correction. Others feel that correction may be needed more than once. It is frequently true that a good predictor needs but a single correction. In this package, we set the upper bound of m to be 3.

It should be noted that when NLMS methods are applied to solve stiff equations, when \mathbf{g} is a function of t alone or is a constant, the PC^m procedure may not be needed.

5.3 SELF-STARTING

The possibility of missing starting values is common with all multistep methods. The missing values have to be provided by an independent method. The Runge-Kutta and Adam-Bashforth methods are often used as starters. The self-starting procedure is provided here by using NLMS or LMS first-order methods. As mentioned previously, NLMS methods allow the use of large step size to solve stiff equations; therefore, we shall adopt the explicit NLMS methods of order one as a starter for solving stiff equations. The Adams-Bashforth first-order method is used as a starter for solving nonstiff equations, which requires the use of a very small step size. In the above case, LMS methods are called for and the self-starting procedure requested. It is suggested that the user specify a small initial step size to achieve satisfactory accuracy.

In most instances, the use of NLMS explicit methods of order one to self-start is recommended because these methods are a good starter and, most of all, because the step size is not restricted; the only requirement is that the \mathbf{g} function must be defined. To do this, a simple modification can be incorporated in the START subroutine. The user can save the indicator, METHOD, from the argument and set it to 1 after entry, then reset METHOD to the original state before RETURN.

5.4 SELECTION OF CHARACTERISTIC COEFFICIENTS α_i

The characteristic polynomial of LMS methods takes the form

$$\rho(\zeta) = \sum_{i=0}^K \alpha_i \zeta^i = 0, \quad \alpha_K = 1. \quad (5.1)$$

Let ζ_j be the roots of (5.1). The relationship

$$\prod_{j=1}^K (\zeta - \zeta_j) = \sum_{i=0}^K \alpha_i \zeta^i \quad (5.2)$$

enables us to express α_i as a function of ζ_j ; the ζ_j determine the root condition and, thus, the stability. In this manner, after the ζ_j are chosen to satisfy the root condition of stability, the α_i can be determined.

For NLMS methods, first we determine $\phi_{Ki}(\mathbf{A}h)$, which are dependent on α_i , as given by formulas (3.14) - (3.28). After $\phi_{Ki}(\mathbf{A}h)$ are determined, the numerical solution is calculated by means of formula (3.2), which is dependent on α_i .

For LMS methods, the α_i are usually tabulated. One should note that, in formula (2.2), if $\mathbf{A} = \mathbf{O}$, NLMS methods reduce to LMS methods. If we multiply (3.9) by $[(\mathbf{A}h)^{j+1}]^{-1}$ and let $\|\mathbf{A}\| \rightarrow 0$, we get

$$\sum_{i=0}^K \alpha_i \frac{1^{j+1}}{(j+1)!} = \frac{1}{j!} \sum_{i=0}^K i^j \phi_{Ki}(\mathbf{O}). \quad (5.3)$$

For $j=0, 1, \dots, p-1$, equation (5.3) gives a formulation of the LMS methods of order p . The matrix form is

$$\begin{pmatrix} 0 & 1 & \dots & K \\ 0 & \frac{1^2}{2!} & \dots & \frac{K^2}{2!} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \frac{1^p}{p!} & \dots & \frac{K^p}{p!} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_K \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 1 & \dots & K \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \frac{1^{p-1}}{(p-1)!} & \dots & \frac{K^{p-1}}{(p-1)!} \end{pmatrix} \begin{pmatrix} \phi_{K0} \\ \phi_{K1} \\ \vdots \\ \phi_{KK} \end{pmatrix}. \quad (5.4)$$

Using (5.4), we can express $\beta_i (= \phi_{Ki}(\mathbf{O}))$ as a function of α_i . We proceed by using the explicit LMS methods of order 2 whose β coefficients are

$$\begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_1/2 \\ \alpha_1/2 + 2\alpha_2 \end{pmatrix}. \quad (5.5)$$

The selection of two roots, 0 and 1, leads to strong stability and gives

$\alpha_0 = 0$, $\alpha_1 = -1$ and $\alpha_2 = 1$. This implies that

$$\begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} -1/2 \\ 3/2 \end{pmatrix},$$

which gives the Adams-Basforth method of order two. The author is unaware of any program that allows a user to select α_1 . One does not

need to make a choice when using tabulated α_1 values. He has found¹ that

the selection of roots of the characteristic polynomial can minimize the initial local discretization error C_{p+1} . Therefore, the selection of α_1 may aid in studying the minimization of C_{p+1} . This is still an open problem.

Hence, this option may not be immediately exercised, but it is available when needed.

Different characteristic roots satisfying the root condition give the characteristic coefficients α_1 . Different families of implicit NLMS methods of order two are selected to solve problem 3 in section 8. These families, their characteristic roots, and the associated characteristic coefficients are identified by the following:

1. Strongly Stable Generalized-Adams Family

characteristic roots: 0, 1

characteristic coefficients: $\alpha_0 = 0$, $\alpha_1 = -1$, $\alpha_2 = 1$.

2. Strongly Stable Family

characteristic roots: 1/2, 1

characteristic coefficients: $\alpha_0 = 1/2$, $\alpha_1 = -3/2$, $\alpha_2 = 1$

3. Weakly Stable Generalized-Milne Family

characteristic roots: -1, 1

characteristic coefficients: $\alpha_0 = -1$, $\alpha_1 = 0$, $\alpha_2 = 1$

5.5 **A AS A FUNCTION OF t**

Consider the initial value problem

$$\mathbf{y}' = \mathbf{A}(t) \mathbf{y} + \mathbf{g}(t, \mathbf{y}); \quad \mathbf{y}(t_0) = \mathbf{y}_0 \quad (5.6)$$

over the interval $[t_0, t_0 + nh]$. We periodically decompose equation (5.6) into

$$\mathbf{y}' = \mathbf{A}(t_i) \mathbf{y} + \bar{\mathbf{g}}(t, \mathbf{y}); \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad (5.7)$$

where

$$\bar{\mathbf{g}}(t, \mathbf{y}) = \left\{ \mathbf{A}(t) - \mathbf{A}(t_i) \right\} \mathbf{y} + \mathbf{g}(t, \mathbf{y}), \quad (5.8)$$

such that $\text{Re} \left\{ \lambda(\mathbf{A}(t_i)) \right\} < 0$. We periodically evaluate $\mathbf{A}(t)$ at $t = t_i$. The requirement that $\text{Re} \left\{ \lambda(\mathbf{A}(t_i)) \right\} < 0$ can be realized by construction and can be assured since $\text{Re} \left\{ \lambda(\mathbf{A}(t)) \right\} < 0$ for all t . The property of $\bar{\mathbf{g}}(t, \mathbf{y})$ can still be maintained for small h since $\|\mathbf{A}(t) - \mathbf{A}(t_i)\|$ is small and can be controlled by the step size. In cases where the user possesses prior knowledge about the problem, he can introduce a constant \mathbf{A} and use the same decomposition technique and perhaps obtain quicker and more precise results. This is not to say decomposition is unique.

5.6 INCLUSION OF LMS METHODS

As a part of this package LMS methods are included, whose step numbers are chosen to be compatible with the same step numbers used for NLMS methods that do not exceed order three. The application of LMS methods is indicated by an indicator, METHOD, which has the following definition:

$$\text{METHOD} = \begin{cases} 0, & \text{LMS methods} \\ 1, & \text{NLMS methods.} \end{cases}$$

The inclusion of LMS methods is merely an option. The reader is asked to consult reference 4 for an efficient implementation of the Adams

methods. However, the LMS methods incorporated in this package cover a complete linear multistep family rather than an Adams family alone.

5.7 MISCELLANEOUS FEATURES

1. For solving stiff equations, if $\mathbf{g}(t, \mathbf{y}) = \mathbf{g}(t)$, i.e., independent of \mathbf{y} , no correction is needed. To exercise this feature, set the indicator IPC equal to zero so that unnecessary computations are avoided.

2. If the stiff equation to be solved is homogeneous and \mathbf{A} is a constant matrix, there is no need to calculate $h \sum_{i=0}^K \phi_{Ki}(\mathbf{A}h) \mathbf{g}_{n+i}$. By setting the

indicator IGFN equal to zero, the program will bypass the $h \sum_{i=0}^K \phi_{Ki}(\mathbf{A}h) \mathbf{g}_{n+i}$

computation loop. In the event that \mathbf{A} is a function of t , even if the system is homogeneous, then $\mathbf{g}(t, \mathbf{y}) = \mathbf{0}$ must be defined in the GFN subroutine.

3. A user's starter can be supplied to replace the START through either the argument or the complete replacement of the subroutine.

Subroutine INVERT can be treated in the same manner as above.

6. PROGRAMMING ASPECTS

6.1 NOTES TO USERS

1. Double precision arithmetic is used for all computations.
2. Subroutines not used by the user can be ignored. They are well-defined internally.
3. Subroutines that are required to be supplied by the user must not be ignored in order to obtain correct computations.
 - a. If NLMS methods are used, GFN must be supplied, except
 $\mathbf{g}(t, y) = \mathbf{0}$ (IGFN = 0).
 - b. If LMS methods are used, FFN must be supplied.
 - c. If \mathbf{A} is a function of t , AFNT must be defined.
4. The solution vector is either in YNEW(I) or Y(KSTEP + 1, I). The associated t_i is in TEA. The user who desires to output his results can incorporate the desired output format into PRINT subroutine.
5. The existing Univac 1108 EXEC 8 library matrix inversion subroutine DGJR is used for test problems.
6. Certain detections to inputs are provided in FORTRAN V programs. Default values are assigned to protect a great number of inputs. This advantage can be used since the inputs are defined on NAMELIST. However, this advantage is not applicable to ANSI FORTRAN users; their inputs must be supplied correctly.
7. Meanings of a few special indicators in DIFEQ:

Indicators, brought to the DIFEQ program through the CALL argument, are IG, IV, INDEX, IAT. These are normal indicators.

IG
IV
INDEX
IAT

Indicates whether or not

$\mathbf{g}(t, y) = \mathbf{0}$.
a variable step size is used.
a method is explicit.
 \mathbf{A} is a function of t .

Other than these indicators, there are a few indicators specially used by DIFEQ to produce computational efficiency and accuracy. Their meanings need to be explained.

LMT. Normal use is to define $LMT = m = 3$ when a variable-step-size, PC^m procedure is used. To ensure this, LMT is set to 3 initially. In the event a fixed-step-size, implicit method is asked for, we make use of this indicator to yield the following control.

- a. When $IV = 0$ and $INDEX = 1$, a fixed-step-size, implicit method is asked for.
- b. After a complete calculation of the required implicit formula, LMT is changed to contain 1. The purpose of this modification is to inform the program to predict the next y_n value in order to implicitly calculate the y_n . This internal modification does not affect the subsequent calculations.

ITER. A counter for every successful corrector's convergence or every fixed-step calculation. To take advantage of this, this indicator is designed to appear in the argument of PRINT, so that the user may use this information to control his printouts. As an example, if the user wants to print out results at every 100 successful calculations, he needs to define a statement in the PRINT program that reads:

```
IF(MOD(ITER,100) .EQ. 0)WRITE(4,12)H,T,(Y(L),L=1,N).
```

ISTEP. If a fixed step size is used to proceed for subsequent computations, the matrix exponentials and the determination of $\phi_{Ki}(\Delta h)$ can be

calculated only once. When a variable step size is used, it is necessary to calculate the matrix exponentials and to determine the $\phi_{Ki}(\Delta h)$ for every h .

When the program finds a successful h that leads to the corrector's convergence as well as satisfying user's tolerance, we take a new approach: We can use the same h to proceed, if required conditions are met, for at most three calculations; we then consider doubling the step size. Suppose we double the step size too soon. Then it is necessary to calculate the matrix exponentials and to determine the $\phi_{Ki}(\Delta h)$. If this new h does not lead to the corrector's convergence, all above computations are wasteful. Then we need somehow to use the matrix exponentials and the $\phi_{Ki}(\Delta h)$ for the old h . Without worrying about memory space, the old values can be saved; without worrying

about computation time, the old values can be recalculated. Certainly, we cannot save both the time and the space simultaneously. Now, the reason that ISTEP is used to minimize cost becomes apparent. In the case of solving stiff equations, chances are good for success, even if h is doubled. The user can take advantage of this by requiring the limit of ISTEP to be 1. The change to the program is almost trivial. The present setup doubles h after each successful calculation.

IMIN. To perform floating-point addition between $T(1)$ and H , the difference between $T(1)$ and H must be within the allowable significant digits of the computer. To maintain m significant digits in floating-point addition, the exponent ($T(1)$) minus the exponent (H) cannot exceed m . We define HMIN to mean h_{\min} and to satisfy this range. In reality, even if the

Lipschitz constant is large and even if the LMS methods are called for, HMIN will not fall outside of the significant digit range, and we can assign a larger h_{\min} to avoid excessive computations. Once this safeguard is built in, when H reaches HMIN more than three times, we will not continue. We will terminate the program with a message for the user to tell him what to do. The variable IMIN is defined to count how many times H reached HMIN.

8. Even though programs work for the same computer in different locations, the input/output (I/O) units may differ in number; users are advised to make sure that the unit numbers are in agreement with the program.

6.2 FORMATS OF USER-SUPPLIED SUBROUTINES

A few optional subroutines must be supplied by the user under certain conditions. These conditions are listed below along with their required formats.

1. INVERT. This is a matrix inversion subroutine, called by NLMS methods whose CALL statement has the format

CALL INVERT (A, N, B),

where

A is an ($n \times n$) matrix

N is the order of matrix A

B contains the A inverse after exit.

TR 5341

As an example, using an existing Univac 1108 library subroutine DGJR (Gauss-Jordan Reduction), the structure of INVERT should look like

```
SUBROUTINE INVERT (A, N, ANS)
PARAMETER NR=20, NC=20
IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION A (NR, NC), ANS (NR, NC)
DIMENSION V (2), JC (NC)
V (1) = 1.D0
DO 1 I = 1, N
DO 1 J = 1, N
1 ANS (I, J) = A (I, J)
CALL DGJR (ANS, NR, NC, N, N, $10, JC, V)
RETURN
10 WRITE (4, 2)
2 FORMAT (3X, 'MATRIX INVERSION ERROR')
RETURN
END
```

2. BEGIN. This subroutine supplies initial values. It is required if the self-starting procedure is not used. Its CALL statement has the format

```
FORTRAN V: CALL BEGIN (K, H, N, Y, YN, YOLD, INVERT,
IT, METHOD, IAT)
ANSI FORTRAN: CALL START (K, H, N, Y, YN, YOLD, IT,
METHOD, IAT),
```

where

K is the step number

H is the step size

N is the order of the system

Y is a two-dimensional array containing the initial values; at most, 4 rows, 20 columns.

YOLD is the same dimension as Y, used as a working storage.

INVERT is the name of a matrix inversion subroutine.

IT is an indicator that indicates whether the calculation of a \mathbf{g} -function is needed.

METHOD indicates use of either NLMS or LMS explicit of order one.

IAT indicates whether \mathbf{A} is a function of t .

As an example, if one uses exact initial values for problem 5, section 8, the structure of BEGIN should look like this in FORTRAN V:

```
SUBROUTINE BEGIN (K, H, N, Y, YN, YOLD, INVERT, IT, METHOD, IAT)
PARAMETER NR = 20, NC = 20
IMPLICIT REAL*8 (A=H, O-Z)
EXTERNAL INVERT
COMMON A (NR, NC), ALPHA (4), T (NC)
DIMENSION Y (4, NC), YN (N), YOLD (4, NC)
H2 = T (1) + H
H3 = H2 + H
Y(2, 1) = DEXP (-H2*H2/2.) - DEXP (-H2) + 1.0
Y(3, 1) = DEXP (-H3*H3/2.) - DEXP (-H3) + 1.0
RETURN
END.
```

In ANSI FORTRAN, the structure should be as follows:

```
SUBROUTINE START (K, H, N, Y, YN, YOLD, IT, METHOD, IAT)
COMMON A (20, 20), ALPHA (4), T (20)
DIMENSION Y (4, 20), YN (20), YOLD (4, 20)
DOUBLE PRECISION A, ALPHA, H, T, Y, YN, YOLD, H2, H3
H2 = T (1) + H
H3 = H2 + H
Y (2, 1) = DEXP (-H2*H2/2.D0) -DEXP (-H2) +1.D0
Y (3, 1) = DEXP (-H3*H3/2.D0) -DEXP (-H3) +1.D0
RETURN
END
```

3. GFN. This subroutine calculates the $\mathbf{g}(t, \mathbf{y})$ at $t = t_i$ for every
1. It is required by NLMS methods only if the differential equation is non-homogeneous. Its CALL statement has the following format:

FORTRAN V: CALL GFN (G, H, N, Y, K, T, A, NR, NC)
ANSI FORTRAN: CALL GFN (G, H, N, Y, K, T, A),

where

G is a vector that contains calculated \mathbf{g} function values

H equals the step size

N is the order of the system

Y is a two-dimensional array Y (I, J): $1 \leq I \leq 4$; $1 \leq J \leq N$

K equals the step number

T is a vector that contains time values

A is an ($n \times n$) matrix

NR equals the number of rows of A

NC is the number of columns of A.

For example, to solve a system of two stiff equations, problem 3 of section 8, the structure of the GFN should be as follows in FORTRAN V:

```
SUBROUTINE GFN (G, H, N, Y, K, T, A, NR, NC)
IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
G (1) = 1.0
G (2) = 1.0
RETURN
END
```

The structure in ANSI FORTRAN should be as follows:

```
SUBROUTINE GFN (G, H, N, Y, K, T, A)
DOUBLE PRECISION A (20, 20), Y (4, 20), G (20), T (20), H
G (1) = 1.D0
G (2) = 1.D0
RETURN
END
```

4. FFN. This FFN subroutine calculates $\mathbf{f}(t, \mathbf{y})$, which is required only by LMS methods. Its CALL statement has the format

CALL FFN (Y, N, FN, I),

where

Y is a 2-dimensional array Y (I, J);

$1 \leq I \leq \text{step number} + 1 ; 1 \leq J \leq N$

N equals the order of the system

FN is a vector to store calculated \mathbf{f} function values

I is the intermediate step index .

For example, to solve a system of two equations, problem 4 of section 8, using LMS methods, the structure of FFN should be as follows for FORTRAN V:

```
SUBROUTINE FFN (Y, N, FN, I)
PARAMETER NR = 20, NC =20
IMPLICIT REAL*8 (A-H, O-Z)
COMMON A (NR, NC), ALPHA (4), T (NC)
DIMENSION FN (1), Y (4, NC)
FN (1) = Y (I, 2)
FN (2) = -Y (I, 1)
RETURN
END
```

For ANSI FORTRAN, the structure is

```
SUBROUTINE FFN (Y, N, FN, I)
COMMON A (20, 20), ALPHA (4), T (20)
DIMENSION FN (20), Y (4, 20)
DOUBLE PRECISION A, ALPHA, FN, T, Y
FN (1) = Y (I, 2)
FN (2) = -Y (I, 1)
RETURN
END
```

TR 5341

5. AFNT. This AFNT subroutine should be supplied by the user when \mathbf{A} is a function of t . This subroutine calculates every $\mathbf{A}(t)$ at $t = t_i$.

Its CALL statement has the format

FORTRAN V: CALL AFNT (A, N, T, NR, NC)
ANSI FORTRAN: CALL AFNT (A, N, T),

where

A is a two-dimensional array, A (NR, NC)

N equals the order of the matrix

T is a time variable

NR is the number of maximum rows of \mathbf{A}

NC equals the number of maximum columns of \mathbf{A} .

For example, to solve a system of one equation (problem 5) using an NLMS method of order 2, the structure of AFNT in FORTRAN V should be:

```
SUBROUTINE AFNT (A, N, T, NR, NC)
DOUBLE PRECISION A (NR, NC), T
A (1, 1) = -T
RETURN
END
```

In ANSI FORTRAN, it should be:

```
SUBROUTINE AFNT (A, N, T)
DOUBLE PRECISION A (20, 20), T
A (1, 1) = -T
RETURN
END
```

6. PRINT. This PRINT is designed for the user's convenience to output the results in his desired formats. Its CALL statement has the format

CALL PRINT (H, T, Y, N, I),

where

H is the present step size

T is the present time step

Y is a one-dimensional array containing the current solution

N is the order of the system

I is an iteration counter.

For example, results are required to be printed out at t = 5 and 10 of problem 5 (section 8) along with exact solutions at the same time values. The PRINT statement can be designed as follows:

```
SUBROUTINE PRINT (H, T, Y, N, I)
DOUBLE PRECISION H, T, Y (1)
12 FORMAT (1X, 2 (E15.8, 2X), 4X, 5 (E15.8, 2X))
    IF (T-4. 995D0) 84,84,83
83 IF (T-5. 005D0) 1184,1184,85
85 IF (T-9. 995D0) 84,84,86
86 IF (T-10. 005D0) 1184,1184,84
1184 WRITE (4,12) H, T, (Y(J), J = 1, N)
    CALL EXACT (T, Y)
84 RETURN
END
```

7. USER'S GUIDE

The following table gives the options, the default conditions, and the input indicators available.

Table 7-1. User's Guide

Option	Default Condition	Input Indicator
A is a function of t	0	IAT \neq 0
ERR	0.1D-11	
F-function (FFN)	0	Called by LMS only if METHOD = 0
final time		TMAX
fixed step size		IPC = 1 and INDEX = { 0 Explicit 1 Implicit}
G-function (GFN)	0	IGFN = { 0 not needed \neq 0 need for stiff eqs.}
INDEX	0	= { 0 Explicit 1 Implicit}
initial time		T (1)
INVERT		user-supplied matrix inversion

Table 7-1. User's Guide (cont)

Option	Default Condition	Input Indicator
KSTEP, step number	1	KSTEP
METHOD	1	= { 0 LMS 1 NLMS
PC ^m	0	IPC = 1 or IAT ≠ 0, m = 3
START	self-starting	User-supplied starter
step size	0.01	H
step size (maximum)	0.1	HMAX
variable step size		IPC = 1
initial vector		YZERO
α_i	GAB-GAM	ALPHA
order of the system		N

8. NUMERICAL RESULTS AND INPUT SETUPS

8.1 PROBLEM IDENTIFICATION

An ordinary differential equation is said to belong to the class (N, S, H, L) if it satisfies the following definitions:

$$\begin{aligned} N &= \begin{cases} 0 & \text{first order} \\ 1 & \text{higher order} \end{cases} \\ S &= \begin{cases} 0 & \text{nonstiff} \\ 1 & \text{stiff} \end{cases} \\ H &= \begin{cases} 0 & \text{homogeneous} \\ 1 & \text{nonhomogeneous} \end{cases} \\ L &= \begin{cases} 0 & \text{linear in} \\ 1 & \text{nonlinear in.} \end{cases} \end{aligned}$$

For example, the equation

$$\mathbf{y}' = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{y}; \quad \mathbf{y}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

is a first-order, nonstiff, homogeneous, linear differential equation. Therefore it belongs to (0, 0, 0, 0).

8.2 SELECTION OF TEST PROBLEMS

Seven test problems have been selected from various sources in order to illustrate the different options. At least one feature is exercised in each problem. The exact start will be used for most test problems. The formats of user-supplied subroutines are described within each problem in FORTRAN V language. The main body of ANSI FORTRAN is the same as FORTRAN V of each subroutine; therefore, the ANSI FORTRAN version is

6-11

omitted. For exact ANSI FORTRAN formats, the user can consult the ANSI FORTRAN in the appendix. Table 8-1 itemizes the seven problems, which are then each discussed in detail.

8.3 PROBLEMS, INPUT SETUPS, AND SOLUTIONS

Problem 1

$$\mathbf{y}' = -100 \mathbf{y} + (1 + t^2); \mathbf{y}(0) = 1.$$

Exact Solution

$$\mathbf{y}(t) = \left(1 - \frac{1}{100} - \frac{2}{100^3}\right) e^{-100t} + \frac{1}{100} + \frac{1}{100^3} (100^2 t^2 - 200t + 2).$$

Classification

$$(0, 1, 1, 0).$$

Eigenvalues of A

$$\{-100\}$$

Method Applied

Implicit NLMS of order 3 with fixed step size. Linear multistep of order 3 with PC^M procedure.

Step Size Used

1.25 and 2.5 for NLMS. Variable step size for LMS with initial h = 0.01.

Time Interval

$$[0, 10]$$

Special Features Exercised

Fixed step size.

Variable step size.

Table 8-1. The Seven Test Problems

Problem No.	Order of System	Class	Eigenvalues	$\Delta(t)$	$\mathbf{g}(t, \mathbf{y})$	Options Applied	Source
1	1	(0, 1, 1, 0)	real		$\mathbf{g}(t)$	Implicit NLMS and LMS of order 3	Lee ¹
2	2	(0, 1, 1, 0)	real	○		Implicit NLMS of order 2	Weaver ¹⁰
3	2	(0, 0, 1, 0)	real		constant	Explicit NLMS of order 3	Lee ¹
4	2	(1, 0, 0, 0)	purely imaginary	○		Explicit LMS of order 2	Krough ⁹
5	1	(0, 0, 1, 0) (0, 1, 1, 0)	real	X	$\mathbf{g}(t)$	NLMS of order 3	Krough ⁹
6	1	(1, 1, 1, 1)	real		$\mathbf{g}(t, \mathbf{y})$	NLMS of order 3	Jain ⁸
7	4	(0, 1, 1, 0)	real		$\mathbf{g}(t)$	NLMS or order 1	Gear, ⁷ Krough ⁹

TR 5341

Special Features Exercised (cont)

Exact start.
PC^m procedure.

FORTRAN V Inputs Set-Up

Fixed step size, implicit NLMS of order 3.

```
$INPUTS
N=1, KSTEP=3, ALPHA(1)=0.D0, 0.D0, -1.D0, 1.D0,
T(1)=0.D0, TMAX=.1D2,
YZERO(1)=1.D0, A(1,1)=-1.D2,
H=1.25D0,
IGFN=1, IPC=0, METHOD=1, INDEX=1,
$END
```

Variable step size, LMS of order 3.

```
$INPUTS
N=1, KSTEP=3, ALPHA(1)=0.D0, 0.D0, -1.D0, 1.D0,
T(1)= 0D0, TMAX=.1D2,
YZERO(1)=1.D0, ERR=.1D -11,
H=.1D-3, HMAX=.1D-2,
IPC=1, METHOD=0, INDEX=0
$END
```

User-Supplied Subroutines

```
SUBROUTINE GFN (G,H,N,Y,J,T,A,NR,NC)
IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
TH = T (1) + (J-1) *H
G (1) = 1. + TH * TH
RETURN
END
```

User-Supplied Subroutines (cont)

```
SUBROUTINE FFN (Y, N, FN, I)
PARAMETER NR=20, NC=20
IMPLICIT REAL*8 (A-H, O-Z)
COMMON A (NR, NC), ALPHA (4), T (NC)
DIMENSION FN (1), Y (4, NC)
FN (1) = -100, *Y (I, 1) + (1.+T (I)**2)
RETURN
END
```

An Exact Starter - BEGIN

```
SUBROUTINE BEGIN (K, H, N, Y, YN, YOLD, INVERT, IT, METHOD, IAT)
PARAMETER NR=20, NC=20
IMPLICIT REAL*8 (A-H, O-Z)
EXTERNAL INVERT
COMMON Y (4, NC), YN (N), YOLD (4, NC)
H2 = T (1) + H
H3 = H2 + H
H4 = H3 + H
X = 1. -. 01-2./100.**3
X4=X*DEXP(-100.*H4)
X3=X*DEXP(-100.*H3)
X=X*DEXP(-100.*H2)
W=(100.*H2)**2-200.*H2+2.
Y(2, 1)=X+.01+W/100.**3
W=(100.*H3)**2-200.*H3+2.
Y(3, 1)=X3+.01+W/100.**3
W=(100.*H4)**2-200.*H4+2.
Y(4, 1)=X4+.01+W/100.**3
RETURN
END
```

SOURCE: Lee¹.

Numerical Results

Give results around $t = 5, 10$.

Implicit NLMS methods of order 3.

t	$\mathbf{y}(t)$
.50000000+01	.25900200-00*
	.25900200-00†
.10000000+02	.10080020+01*
	.10080020+01†

LMS methods of order 3

t	$\mathbf{y}(t)$
.50047500+01	.25497628-00*
	.25497628-00†
.10020750+02	.10121522+01*
	.10121522+01†

Remarks

Nonlinear multistep methods can arrive at $t = 10$ in one step by using $h = 2.5$ at the cost of 4.1 seconds CPU time. However, in order to obtain $\mathbf{y}(5)$, we use $h = 1.25$; the CPU time costs a little more—4.4 seconds. These NLMS methods are designed to allow the use of large step size and to be effective for those $\mathbf{g}(t, \mathbf{y})$ belonging to the class of low-order polynomials. Therefore, it is not surprising that the implicit NLMS method of order 3 produces accurate results.

*Solution produced by the methods

†Exact solution

Problem 2

$$\mathbf{y}' = \begin{pmatrix} \frac{\delta k - \beta}{l} & \lambda_1 \\ \frac{\beta_1}{l} & -\lambda_1 \end{pmatrix} \mathbf{y}; \mathbf{y}(0) = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

using $\delta k = 1.0075$

$$\beta = \beta_1 = 0.0075$$

$$\lambda_1 = 0.075$$

$$l = 10^{-6}$$

Exact Solution

$$\mathbf{y}(2) = (-.6414 \ 1073-07, -.8552 \ 1424 + 00)^T$$

$$\mathbf{y}(5) = (-.5130 \ 4190-07, -.6840 \ 5581 + 00)^T$$

$$\mathbf{y}(10) = (-.3536 \ 0130-07, -.4714 \ 6837 + 00)^T$$

Classification

$$(0, 1, 0, 0)$$

Eigenvalues of A

$$\{-0.0744375, -10^6\}$$

Method Applied

Implicit NLMS of order 2 with fixed step size.

Step Size Used

$$1.0$$

Time Interval

$$[0, 10]$$

Special Features Exercised

Fixed step size
Exact Solution

FORTRAN V Inputs Setup

Fixed step size, implicit NLMS of order 2

\$INPUTS

N = 2, KSTEP = 2, ALPHA (1) = 0.D0, -1.D0, 1.D0
 T (1) = 0.D0, TMAX = 10.D0
 YZERO (1) = 1.D0, -1.D0, H=1.D0
 IGFN = 0, IPC = 0, METHOD = 1, INDEX = 1
 \$END

User-Supplied Subroutine

Since $\mathbf{g}(t, \mathbf{y}) = \mathbf{0}$, IGFN is set to 0. Hence, the GFN is not called for.

Source Weaver¹⁰

The infinite-medium reactor kinetic equations, in standard form, with m delayed neutron groups, can be expressed as

$$\frac{dn}{dt} = \frac{\delta k - \beta}{\ell} n + \sum_{i=0}^m \lambda_i C_i$$

$$\frac{dC_i}{dt} = \frac{\beta_i}{\ell} n - \lambda_i C_i,$$

where

β_i = the fraction of the total number of fission neutrons belonging to the i^{th} group precursor that are delayed,

β = the fraction of the fission neutrons that are delayed,

ℓ = neutron generation time (sec).

λ_i = decay constant for the i^{th} group precursor,

n = neutron density,

C_i = the concentration of the i^{th} precursor, and

δ_k = a constant for a reactor with a constant excess of reactivity.

Considering a single delayed neutron group, we obtain

$$\begin{pmatrix} \frac{dn}{dt} \\ \frac{dC_1}{dt} \end{pmatrix} - \begin{pmatrix} \frac{\delta k - \beta}{\ell} & \lambda_1 \\ \frac{\beta}{\ell} & -\lambda_1 \end{pmatrix} \begin{pmatrix} n \\ C_1 \end{pmatrix}$$

Numerical Results

Give results at $t = 2, 5, 10$.

t	$y_1(t)$	$y_2(t)$
.2000 0000 + 01	-.6414 1073-07 -.6414 1073-07	-.8552 1424+00* -.8552 1424+00†
.5000 0000+01	-.5130 4190-07 -.5130 4190-07	-.6840 5581+00 -.6840 5581+00
.1000 0000+02	-.3536 0130-07 -.3536 0130-07	-.4714 6837+00 -.4714 6837+00

Remarks

A-stable NLMS methods can solve homogeneous stiff equations exactly in the absence of round-off errors. An extremely small step size is required for the same accuracy ($.1 \times 10^{-6}$) if LMS methods are used.

*Produced by implicit NLMS methods of order 2

†Exact solution

Problem 3

$$\mathbf{y}' = \begin{pmatrix} 0 & 1 \\ 10 & -9 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix}; \quad \mathbf{y}(0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

Exact Solution

$$\mathbf{y}(t) = \begin{pmatrix} 2e^t - 1 \\ 2e^t - 1 \end{pmatrix}$$

Classification

$$(0, 0, 1, 0)$$

Eigenvalues of A

$$\{-10, 1\}$$

Methods Applied

Explicit NLMS method of order 3.

Step Size Used

0.1 and 1.0.

Time Interval

$$[0, 10]$$

Special Features Exercised

Fixed step size.

Self-start.

FORTRAN V Inputs Set-Up

Fixed step size, explicit NLMS of order 3.

```
$INPUTS
N=2, KSTEP=3, ALPHA (1)=0.D0, 0.D0, -1.D0, 1.D0,
T(1)=0.D0, TMAX=.1D2,
YZERO(1)=1.D0, 1.D0,
A(1,1)=0.D0, A(1,2)=1.D0, A(2,1)=.1D2, A(2,2)=-9.D0,
H=1.D0,
IGFN=1, IPC=0, METHOD=1, INDEX=0
$END
```

User-Supplied Subroutines

```
SUBROUTINE GFN (G,H,N,Y,J,T,A,NR,NC)
IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
G (1) = 1.0
G (2) = 1.0
RETURN
END
```

Source: Lee¹.

Numerical Results

Gives results at t=1, 5, and 10.

t	y ₁ (t)	y ₂ (t)
.10000000+01	.44365637+01 .44365637+01	.44365637+01* .44365637+01†
.50000000+01	.29582632+03 .29582632+03	.29582632+03 .29582632+03
.10000000+02	.44051932+05 .44051932+05	.44051932+05 .44051932+05

*Produced by explicit NLMS methods of order 3.

†Exact solution.

TR 5341

Since LMS methods are not used, FFN is not called for. The built-in self-starter is used.

Problem 4

$$u'' + u = 0; u(0) = 0, u'(0) = 1.$$

Equivalent Problem

$$\begin{pmatrix} u' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix}; \begin{pmatrix} u(0) \\ y(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Exact Solution

$$u = \sin t.$$

Classification

$$(1, 0, 0, 0)$$

Eigenvalues of \mathbf{A}

$$\{ \pm i \} .$$

Method Applied

LMS methods of order 2 with PC^m procedure.

Step Size Used

Initial step size = π .

Remarks

In the absence of round-off errors, NLMS methods of different orders are equivalently accurate if $\mathbf{g}(t, \mathbf{y})$ is a constant; NLMS methods of any order should produce accurate results for this problem.

Time Interval
 $[0, 6\pi]$.
Special Features Exercised

Variable step size

PC^m procedure

Exact start.

FORTRAN V Inputs Set-Up

\$INPUTS

N=2, KSTEP=2, IPC=1, INDEX=0, METHOD=0

T(1)=0.D0, TMAX=18.75D0

YZERO(1)=0.D0, 1.D0, ERR=.1D-11

ALPHA(1) = 0.D0, -1.D0, 1.D0

\$END

In MAIN, H is defined to be π initially and HMAX = H/16.User-Supplied Subroutines

SUBROUTINE FFN (Y, N, FN, I)

COMMON A (20, 20), ALPHA (4), T (20)

DIMENSION FN(20, Y(4,20))

FN (1) = Y (I, 2)

FN (2) = -Y (I, 1)

RETURN

END

Source: Krough⁹Numerical Results

Give maximum absolute error at t = 2, 4, 6.

t	Maximum Absolute Error
2 π	.2307 1215-12
4 π	.2307 0955-12
6 π	.2307 1128-12

TR 5341

Remarks

This problem demonstrates that high-order differential equations can be decomposed into a system of first-order equations so that this package is applicable. Since this is not a stiff equation, we choose the variable-step-size and PC^m procedure to examine the efficiency of the LMS portion.

Problem 5

$$\mathbf{y}' = t(1 - \mathbf{y}) + (1-t)e^{-t}; \mathbf{y}(.1) = 1.0901751.$$

Equivalent Problem

$$\mathbf{y}' = -t\mathbf{y} + t + (1-t)e^{-t}; \mathbf{y}(.1) = 1.0901751.$$

Exact Solution

$$\mathbf{y}(t) = e^{-\frac{t^2}{2}} - e^{-t} + 1.$$

Classification

$$0 < t \leq 1 \quad (0, 0, 1, 0)$$

$$1 < t \quad (0, 1, 1, 0).$$

Eigenvalues of A

$$\{-t\}.$$

Methods Applied

NLMS methods of order 3.

Step Size Used

Variable step size with initial h = 0.01.

Time Interval

$$[0.1, 50.0].$$

Special Features Exercised

Automatic periodic decomposition of $\mathbf{A}(t)$.

PC^m procedure.

Variable-step-size procedure.

FORTRAN V Inputs Set-Up

Variable step size, NLMS of order 3

\$INPUTS

```
N=1, KSTEP=3, ALPHA(1)=0.D0, 0.D0, -1.D0, 1.D0,  
T(1)=.1D0, TMAX=.5D2,  
YZERO(1)=1.0901751D0, ERR=.1D-11,  
H=.1D-1, HMAX=.1D0,  
IGFN=1, IPC=1, METHOD=1, INDEX=0, IAT=1  
$END
```

User-Supplied Subroutines

```
SUBROUTINE GFN (G,H,N,Y,J,T,A,NR,NC)  
IMPLICIT REAL*8 (A-H, O-Z)  
DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)  
G (1) = T (J) + (1.-T (J))*DEXP(-T (J))  
RETURN  
END
```

```
SUBROUTINE AFNT (A, N, T, NR, NC)  
DOUBLE PRECISION A (NR, NC), T  
A (1,1) = -T  
RETURN  
END
```

TR 5341

An Exact Starter - BEGIN

```
SUBROUTINE BEGIN (K, H, N, Y, YN, YOLD, INVERT, IT, METHOD, IAT)
PARAMETER NR=20, NC=20
IMPLICIT REAL*8 (A-H, O-Z)
EXTERNAL INVERT
COMMON A (NR, NC), ALPHA (4), T(NC)
DIMENSION Y (4, NC), YN (N), YOLD (4, NC)
DO 1 I = 2, 4
T (I) = T (I-1) + H
1 Y (I, 1) = DEXP (-T (I) * T (I)/2.) -DEXP (-T (I)) + 1.0
RETURN
END
```

Source: Krough⁹.

Numerical Results

Give results at t = 1.0, 10, and 30.

t	y (t)
.10000000+01	.12386512-01*
	.12386512-01†
.10000000+02	.99995460+00
	.99995460+00
.30000000+02	.10000000+01
	.10000000+01

*Produced by NLMS methods of order 3 .

†Exact solution

Problem 6

$$\mathbf{y}' = -100t \mathbf{y}^2; \quad \mathbf{y}(1) = \frac{1}{51}$$

Equivalent Problem

$$\mathbf{y}' = -100 \mathbf{y} + 100 \mathbf{y}(1-t\mathbf{y}); \quad \mathbf{y}(1) = \frac{1}{51}$$

Exact Solution

$$\mathbf{y}(t) = \frac{1}{1 + 50t^2}$$

Classification

(0, 1, 1, 1)

Eigenvalues of A

{ -100 } .

Methods Applied

NLMS methods of order 3.

Step Size Used

Variable step size with initial $h = 0.01$.

Time Interval

[1, 50] .

Special Features Exercised

Variable step size

$\mathbf{g} = \mathbf{g}(t, \mathbf{y})$,

Exact start.

PC^m procedure.

FORTRAN V Inputs Set-up

Variable step size , NLMS order of 3.

\$INPUTS

```
N=1, KSTEP=3, ALPHA(1)=0.D0, 0.D0, -1.D0, 1.D0,  
T(1)=1.D0, TMAX=.5D2,  
YZERO(1)=.19607843D-1, ERR=.1D-11,  
A (1,1) = -100.D0,  
H = .1D-1, HMAX = .1D0,  
IGFN = 1, IPC = 1, METHOD = 1, INDEX = 0  
$END
```

User-Supplied Subroutines

```
SUBROUTINE GFN (G,H,N,Y,J,T,A,NR,NC)  
IMPLICIT REAL *8 (A-H, O-Z)  
DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)  
G (1) = 100.*Y (J, 1) * (1.-T (J) * Y (J, 1))  
RETURN  
END
```

An Exact Starter: BEGIN

```
SUBROUTINE BEGIN (K, H, N, Y, YN, YOLD, INVERT, IT, METHOD, IAT)  
PARAMETER NR=20, NC=20  
IMPLICIT REAL*1 (A-H, O-Z)  
EXTERNAL INVERT  
COMMON A (NR, NC), ALPHA (4), T (NC)  
DIMENSION Y (4, NC), YN (N), YOLD (4 (NC))  
DO 1 I = 2,4  
T (I) = T (I-1) + H  
1 Y (I, 1) = 1./(1.+50.*T (I))  
RETURN  
END
```

Source: Jain ⁸.

Numerical Results

Give results around $t = 5, 10, 20, 30, and 50.$

.5003 1250 +01	.7983 6304-03*
	.7983 6304-03†
.1000 1875 +02	.19988506-03
	.19988506-03
.2000 6875 +02	.49963146-04
	.49963146-04
.3000 4375 +02	.22215249-04
	.22215249-04
.5001 4375 +02	.7995 3381-05
	.7995 3381-05

Remarks

The decomposition technique is applied successfully with the PC^m procedure for this nonlinear problem. Here, $\mathbf{g}(t, \mathbf{y})$ is not a low-order polynomial, but the program can be seen to vary the step size automatically and to maintain convergence and reasonable accuracy.

Problem 7

$$\mathbf{y}' = \mathbf{UZ} - \mathbf{UBUy} ; \quad \mathbf{y}(0) = (-1, -1, -1, -1)^T$$

where

$$\begin{aligned}\mathbf{U} &= \frac{1}{2} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \\ \mathbf{Z} &= \left(w_1^2, w_2^2, w_3^2, w_4^2 \right)^T\end{aligned}$$

*Solution produced by NLMS methods

†Exact solution.

$$\mathbf{W} = \begin{pmatrix} w_1, w_2, w_3, w_4 \end{pmatrix}^T = \mathbf{U} \mathbf{y}$$

$$\mathbf{B} = \text{diag}(\beta_1, \beta_2, \beta_3, \beta_4)$$

$$\beta_1 = 1000, \beta_2 = 800, \beta_3 = -10, \beta_4 = 0.001.$$

Exact Solution

$$\mathbf{y}(t) = \mathbf{U} \left(\begin{array}{c} \frac{\beta_1}{1 - (1 + \beta_1) e^{-\beta_1 t}}, \quad \frac{\beta_2}{1 - (1 + \beta_2) e^{-\beta_2 t}} \\ \frac{\beta_3}{1 - (1 + \beta_3) e^{-\beta_3 t}}, \quad \frac{\beta_4}{1 - (1 + \beta_4) e^{-\beta_4 t}} \end{array} \right)^T$$

Classification

$$(0, 1, 1, 0).$$

Eigenvalues of \mathbf{A}

$$\{-1002, -802, 8, -2.001\} \quad \text{initially}$$

$$\{-1000, -800, -10, -0.001\} \quad \text{when } 0.001t \geq 1.$$

Method Applied

NLMS methods of order 1

Step Size Used

$$0.01 \leq t \leq 1, h = .1D-2.$$

$$1 < t \leq 10, h = .1D-1,$$

$$10 < t \leq 1000, h = .1D0.$$

Time Interval

$$[0.01, 100]$$

Special Features

Fixed step size.

FORTRAN V Inputs Set-Up

To maintain better precision, matrix \mathbf{A} ($A(I,J)$) and vector \mathbf{y}_0 ($YZERO$) are calculated by programs specially incorporated in MAIN.

Fixed step size, NLMS of order 1.

```
$INPUTS
N = 4, KSTEP = 1, ALPHA (1) = -.1D0, 1.D0,
T (1) = .1D-1, TMAX = 1.D3
H = .1D-2,
IGFN = 1, IPC = 0, METHOD = 1, INDEX = 0
$END
```

User-Supplied Subroutines

```
SUBROUTINE GFN (G, H, N, Y, J, T, A, NR, NC)
IMPLICIT REA*8 (A-H, O-Z)
DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
DIMENSION W (4)
WW = 0.0
D0 10 K = 1, N
10 WW = WW + Y (J, K)
D0 5 I = 1, N
W (I) = WW - 2.*Y (J, I)
5 W (I) = W (I) *W (I) / 4.0
G (1) = (-W(1) + W(2) + W(3) + W(4))/2.0
G (2) = ( W(1) - W(2) + W(3) + W(4))/2.0
G (3) = ( W(1) + W(2) - W(3) + W(4))/2.0
G (4) = ( W(1) + W(2) + W(3) - W(4))/2.0
RETURN
END
```

Source: Krough, Gear⁷.

Numerical Results

Given results at t = 50, 500, 1000

t

50	-.50095236+01 -.50095561+01	-.50095326+01 -.50095561+01	.49904764+01 .49904439+01	-.49904764+01* -.49904439+01†
500	-.50007681+01 -.50007688+01	-.50007681+01 -.50007688+01	.49992319+01 .49992312+01	-.49992319+01 -.49992312+01
1000	-.50002903+01 -.50002905+01	-.50002903+01 -.50002905+01	.49997097+01 .49997095+01	-.49997097+01 -.49997095+01

Remarks

Since only three different step sizes are used, there are only three matrix inversions and three matrix exponentials needed. To reach t = 1000 takes about 4 min 46 sec with a maximum error $\sim .38 \times 10^{-7}$.

*Solution produced by NLMS methods.

†Exact solution.

9. CONCLUSIONS

A family of strongly stable and consistent NLMS methods has been developed. When applying NLMS methods to the problem $\mathbf{y}' = \mathbf{A}\mathbf{y}$, which implies $\mathbf{g}(t, \mathbf{y}) = \mathbf{0}$, NLMS methods produce the approximate solution $\mathbf{y}_n = e^{\mathbf{A}t_n} \mathbf{y}_0 = e^{n\mathbf{A}h} \mathbf{y}_0$. Since the matrix exponential is computed by the Pade approximation,^{12,13} which is stable, the $\lim \| \mathbf{y}_n \| \rightarrow 0$ as $n \rightarrow \infty$ establishing the A-stability in the sense of Dahlquist.¹³

If $\mathbf{g}(t, \mathbf{y})$ is a slowly varying function that can be approximated by a low-order polynomial in t and \mathbf{A} is a slowly varying function in t , NLMS methods allow the use of a larger step size and can produce almost exact solution in the absence of round-off errors. In the event that $\mathbf{g}(t, \mathbf{y})$ is not a low-order polynomial in t or not a slowly varying function in t , NLMS methods can still produce accurate results if not too large a step size is used.

In general, because of the initial local discretization error, implicit NLMS methods are better than explicit NLMS methods. High-order NLMS methods are better than low-order NLMS methods. In the event that

$\mathbf{g}(t, \mathbf{y})$ is a constant in t and \mathbf{y} , we see that

$$\sum_{i=0}^{K-1} \phi_{Ki}^{(E)} = \sum_{i=0}^K \phi_{Ki}^{(I)}$$

Therefore, explicit NLMS of order K is equivalent to implicit NLMS of order K . In this case, low-order NLMS methods will suffice; high-order NLMS methods are not necessary. In fact, high-order NLMS methods require more computations, a good Pade approximation, and an accurate matrix inversion.

Several numerical experiments have been performed upon test problems utilizing different selection of α_i , from both strongly and weakly stable NLMS families. The numerical results agree to nine significant digits.

TR 5341

The numerical results presented in this report mainly are used to demonstrate that the NLMS program can produce satisfactory results. The computations herein were intended for illustrative purposes only. It is our intention to apply NLMS methods to more difficult problems. We intend to develop criteria to determine how good NLMS methods are. These will be reported separately.

The NLMS package requires the user to guess an initial step size h . However, the package is "robust"; it determines the most favorable step size for rapid computation. Experience indicates that less total computation time results if h is chosen too small, rather than too large.

If a large, stiff system is to be solved with the variable-step-size technique, an eigenvalue-eigenvector technique such as EISPACK is recommended to replace PADE. This can save a large amount of computation time.

10. REFERENCES

1. D. Lee, Nonlinear Multistep Methods for Solving Initial Value Problems in Ordinary Differential Equations, Ph. D. dissertation, Polytechnic Institute of New York, 1974. Also NUSC Technical Report 4775, 24 May 1974.
2. P. Henrici, Discrete Variable Methods in Ordinary Differential Equations, John Wiley and Sons, Inc., New York, 1962.
3. H. B. Keller, Numerical Methods for Two- Point Boundary-Value Problems, Blaisdell Publishing Co., Waltham, Massachusetts, 1968.
4. J. Certaine, "The Solution of Ordinary Differential Equations With Large Time Constants," in Mathematical Methods for Digital Computers, A. Ralston and H. Wilf, eds., John Wiley and Sons, Inc., New York, 1960, pp. 128-132.
5. L. F. Shampine and M. K. Gordon, Computer Solution of Ordinary Differential Equations, W. H. Freeman and Co., San Francisco, 1975.
6. B. L. Ehle, A Comparison of Numerical Methods for Solving Certain Stiff Ordinary Differential Equations, Department of Mathematics, University of Victoria, Canada, Report 70, 1972.
7. C. W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.
8. R. K. Jain, "Some A-Stable Methods for Stiff Ordinary Differential Equations," Mathematics of Computation, vol. 26, no. 117, pp. 71-77, 1972.
9. F. T. Krough, "On Testing a Subroutine for the Numerical Integration of Ordinary Differential Equations," Journal of the ACM, vol. 20, pp. 545-562, 1973.
10. L. E. Weaver, Systems Analysis of Nuclear Reactor Dynamics, Rowman and Littlefield, Inc., New York, 1963.

7
TR 5341

11. R.A. Willoughby, Stiff Differential Systems, Plenum Press, New York and London, 1974.
12. J.L. Blue, and H.K. Gummel, "Rational Approximations to Matrix Exponential for Systems of Stiff Equations," Journal of Computational Physics, vol. 5, 1970, pp. 70-83.
13. G. Dahlquist, "A Special Stability Problem for Linear Multistep Methods," BIT, vol. 3, 1963, pp. 27-43.

TR 5341

Appendix

NLMS COMPUTER PROGRAMS

FORTRAN V

TR 5341

```

1      **** THIS PROGRAM COMMANDS A SET OF SUBROUTINES TO INTEGRATE A
2      **** SYSTEM OF FIRST-ORDER ORDINARY DIFFERENTIAL EQUATIONS OF
3      **** THE FORM
4      ****      DY(T)/DT = A(Y) + G(T,Y)
5      ****      OR
6      ****      DY(T)/DT = F(T,Y)
7      **** OVER THE CLOSED INTERVAL -(T(1),TMAX). THIS IS A FIXED-ORDERING.
8      **** VARIABLE (FIXED AS WELL)-STEP-SIZE PROGRAM. MOST INPUTS ARE
9      **** DETECTED BY ASSIGNING REASONABLE DEFAULT VALUES. ONLY DIFTE IS
10     **** CALLED BY THIS PROGRAM. OTHER SUBROUTINES ARE CALLED BY
11     **** SUBROUTINES. UNUSED SUBROUTINES ARE DEFINED DUMMIES TO SATISFY
12     **** THE COMPILER. ALL COMPUTATIONS ARE PERFORMED IN DOUBLE-PRECISION.
13     **** INPUT PARAMETERS APPEAR ON THE NAMELIST HAVE THE FOLLOWING
14     **** DEFINITIONS
15
16     ****      A      A 2-DIMENSIONAL ARRAY OF (NR X NC) FOR STORING
17     ****      NR     ELEMENTS OF MATRIX A
18
19     ****      ALPHA   CHARACTERISTIC POLYNOMIAL COEFFICIENTS. NO CONCERN TO
20     ****      NC     THE USER. PROGRAM SELECTS ADAMS COEFFICIENTS FOR
21     ****      R      STRONG STABILITY
22
23     ****      TOL    USER-SUPPLIED TOLERANCE. DEFAULT VALUE 0.11D-11
24
25     ****      H      USER-SUGGESTED INITIAL STEP SIZE. DEFAULT VALUE 0.001
26
27     ****      HMAX   THE MAXIMUM STEP SIZE THE USER ALLOWS THE PROGRAM TO
28     ****      CONSIDER
29
30     ****      IAT    SET =0, IF A IS A CONSTANT
31     ****      SET    SET =1, IF A IS A FUNCTION OF T
32
33
34     ****      LGF1   SET =0, IF G(I,Y) IS ABSENT
35     ****      SET    SET =1, IF G(I,Y) IS PRESENT
36
37

```

```

C*      INDX   SET =0 FOR EXPLICIT. SET =1 FOR IMPLICIT
38      C*      SET =0 FOR PREDICTOR OR CORRECTOR
39      C*      SET =1 FOR PREDICTOR-AND-CORRECTOR
40      C*      STEP NUMBER (LESS THAN 4)
41      C*      METHOD SET =0, LNS METHODS
42      C*      SET =1, NLNS METHODS
43      C*      NUMBER OF EQUATIONS
44      C*      A 1-DIMENSIONAL ARRAY. T(1) CONTAINS INITIAL TIME
45      C*      FINAL TIME, ASSIGNED BY THE USER
46      C*      ZERO  A 1-DIMENSIONAL ARRAY, CONTAINING INITIAL Y
47      C*      START A SELF-START SUBROUTINE
48      C*      BEGIN USER-SUPPLIED STARTUP (OPTIONAL)
49      C*      END  A BUILT-IN MATRIX INVERSION SUBROUTINE
50      C*      INVERT A USER-SUPPLIED MATRIX INVERSION (OPTIONAL)
51      C*      START,END,END,END,INVERT ALL ENTER ARGUMENTS OF CALL DIFEQ
52      C*      INPUT SET UP AS FOLLOWS--
```

```

53      C*      INPUTS
54      C*      USER, INPUT DATA
55      C*      SEND
56      C*      *****
```

```

57      C*      PARAMETER NK=20, NC=20
58      C*      IMPLICIT REAL#0 (ATH,UTL)
59      C*      EXTERNAL START,BEGIN,INVERT
60      C*      COMMON A(IJ,N),ALPHA(4),T(NC)
61      C*      DIMENSION T(4,NC),Y(2,NC)
62      C*
63      C*
64      C*
65      C*
66      C*
67      C*
68      C*
69      C*
70      C*
71      C*
```

```

72      DATA H,HMAX,ALPHA/.10-2,.100,4*1.D3/
73      DATA KSTEP,METHOD,INDEX/2*1.0/
74      * NAMELIST /INPUTS/A,ALPHA,ERR,H,HMAX,IAT,IGFN,INDEX,IPC,KSTEP,
75      *          METHOD,N,T,TMAX,YZERO
76      10 READ(3,INPUTS,ERR=100,END=101)
77      CALL DIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,YZERO,BEGIN,INVERT,IGFN,
78      *          END)
79      60 TC 10
80      100 STOP 100
81      101 STOP 101
82      END

```

A-4

```

SUBROUTINE DIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,Y,YZERO,START,INVERT,
*          IG,IV,METHOD,INDEX,IAT)
C*****DIFEQ IS CALLED BY MAIN PROGRAM WHOSE ARGUMENTS ARE ALREADY
C* UNDEFINED IN THE MAIN PROGRAM WHERE IG=IGFN, IV=IPC
C* DIFEQ CALLS 4 SUBROUTINES
C*   START(KSTEP,...,IAT) --- A STARTER
C*   LMS(KSTEP,...,I) --- LINEAR MULTISTEP
C*   NLMS(KSTEP,...,IAT) --- NONLINEAR MULTISTEP
C*   PRINT(TEA,YNEW) --- FOR USER TO PRINTOUT RESULTS
C* SOLUTION VECTOR Y(I) IS YNEW(I) OR Y(KSTEP+1,I)
C* BASED ON USER-SUPPLIED INPUTS, DIFEQ SETS UP THE ITERATIVE
C* PROCEDURE, CONTROLS STARTER, STEP-SIZE CHANGES, PREDICTOR-
C* CORRECTOR, CORRECTOR'S CONVERGENCE, PRINTOUTS AND CALLS FOR
C* THE REQUIRED METHODS
C* PARAMETER NR=20, NC=20
C* IMPLICIT REAL*B (A-H,J-Z)
C* EXTERNAL INVERT,START
C* COMMON A(NR,NC),ALPHA(4),T(NC)

```

```

24      DIMENSION G(NC),Y(4,NC),YIN(NC),YNEW(NC),YOLD(4,NC),YZERO(NC)
25      DATA HMIN/.1D-11/
26      LM1=3
27      WRITE (4,10)
28      10 FORMAT(1H1)
29      *****
30      C* SELECTS ADAMS ALPHA IF ALPHA ARE NOT GIVEN
31      C* *****
32      AF(ALPHA(1) - 10.0) 37.
33      DO 36 I=1,KSTEP
34      36 ALPHA(I)=0.0
35      ALPHA(KSTEP)=-1.0
36      ALPHA(KSTEP+1)=1.0
37      DO 40 I=1,N
38      40 Y(1,I)=YZERO(I)
39      ITER=0
40      ISSTEP=0
41      IZERO=IT(1)
42      ITER=ITER+1
43      IH=0
44      IMIN=0
45      50 CONTINUE
46      *****
47      C* EVALUATE A(T) AT T=T(0) IF IAT NOT 0
48      C* IF KSTEP GT 1, CALLS START
49      *****
50      IF(IAT.NE.0) CALL AFNT(A,N,T(1),NR,NC)
51      IF(KSTEP.EQ.1, AND. INDEX.EQ.0) GO TO 60
52      CALL STAKT(KSTEP,H,N,Y,YIN,YOLD,INVERT,IG,METHOD,IAT)
53      TEA=IT(1)+KSTEP*H
54      IH=IH+1
55      DO 61 J=1,KSTEP
56      61 T(J+1)=T(J)+H

```

```

57      DO 62 J=0,KSTEP
58      DO 62 I=1,N
59      62 YOLD(J+1,I)=Y(J+1,I)
C*****METHOD=0 -- LMS, METHOD=1 -- NLMS
C*      FIXED-STEP SIZE          IV=0, INDEX#0  PREDICTOR
C*      VARIABLE-STEP-SIZE      IV=0, INDEX#1  CORRECTOR
C*      IF(METHOD.NE.0) GO TO 64
60      IF(IV.EQ.0) CALL LMS(KSTEP,H,YOLD,N,YN,0)
61      IF(IV.EQ.0.AND,INDEX.EQ.1) CALL LMS(KSTEP,H,YOLD,N,YN,1)
62      GO TO 59
63      IF(IV.EQ.0.AND,INDEX.EQ.0) CALL NLMS(KSTEP,H,YOLD,N,YN,0,1H,
64      5         INVERT,IG,IAT)
65      IF(IV.EQ.0.AND,INDEX.NE.0) CALL NLMS(KSTEP,H,YOLD,N,YN,1,1H,
66      S         INVERT,IG,IAT)
67      IF(IV.NE.0) CALL NLMS(KSTEP,H,YOLD,N,YN,0,1,INVERT,IG,IAT)
68      59 DO 66 I=1,N
69      66 YOLD(KSTEP+1,I)=YN(I)
70      IF(IV.NE.0 .OR., IAT.NE.0) GO TO 69
71      IF(ILMT .LT. 1) GO TO 69
72      DC 58 I=1,N
73      68 YNEW(I)=YN(I)
74      GO TO 82
C*****CORRECTOR AT MOST CORRECT 3 TIMES
C*      STEP-SIZE CHANGED BY A FACTOR OF 2
C*****ICORR=0
C*****ICORR=ICORR+1
C*      IF(ICORR .LE. LMT) GO TO 75
75      H=H/2.0
76      IF(H .LT. HMIN) IMIN=IMIN+1
77      IF(H .LT. HMIN) H=HMIN
78      IF(IMIN .GT. 3) WRITE (4,1170)
79
80
81
82
83
84
85
86
87
88
89
90
91
92

```

```

93      FORMAT(3X,'H REACHED HMIN, NO CONVERGENCE POSSIBLE')
94      IF(LIMIN .GT. 3) STOP
95      T(1)=TZERO
96      DO 71 I=1,N
97      Y(1,I)=YZERO(I)
98      GO TO 50
99      75 IF(METHOD.EQ.0) CALL LMS(KSTEP,H,Y0,D,N,YNEW,1)
100     IF(METHOD .GT. 0) CALL NLMS(KSTEP,H,YOLD,N,YNEW,1,ICORR,INVENT,
101        *16,IAT)
102     IF(ILNT .NE. 1) GO TO 76
103     DO 77 I=1,N
104     YNEW(I)=YN(I)
105     GO TO 82
106     CONTINUE
107     ANORM=0.0
C***** C* TEST CORRECTOR'S CONVERGENCE. USING MAX NORM
C***** C* TEST CORRECTOR'S CONVERGENCE. USING MAX NORM
108     DO 80 J=1,N
109     G(J)=(YOLD(KSTEP+1,J)-YNEW(J))/YOLD(KSTEP+1,J)
110
111     112     IF(ANORM .GT. DABS(G(J))) GO TO 89
113
114     ANORM=BS(G(J))
115     CONTINUE
116     IF(ANORM - ERR) 82,82,
117     DO 81 J=1,N
118     YOLD(KSTEP+1,J)=YNEW(J)
119     GO TO 70
120     82 DO 83 I=1,N
121     83 Y(KSTEP+1,I)=YNEW(I)
C***** C* RESULTS Y(TEA) IN YNEW(I) AND Y(KSTEP+1,I)
122
123     C* CALL PRINT FOR USER TO PRINT OUT RESULTS
124
125     C* CALL PRINT(H,TEA,YNEW,N,ITER)
126
127     84 CONTINUE

```

TR 5341

```
128 IF(TEA .GT. TMAX) RETURN
129 DO 85 J=1,KSTEP
130   DO 85 I=1,N
131     Y(J,I)=Y(J+1,I)
132     IF(J.EQ.1) YZERO(I)=Y(J,I)
133   CONTINUE
134   T(1)=T(2)
135   TZERO=T(1)
136   IF(IV.EQ.0 .AND. INDEX.EQ.1) LMT=1
137   IF(IV.EQ.0 .AND. INDEX.EQ.1) INDEX=0
138   IF(ILMT .EQ. 1) IH=0
139   IF(IV .EQ. 0) GO TO 60
140   T(1)=TEA
141   TZERO=T(1)
142   DO 86 I=1,N
143     YZERO(I)=Y(KSTEP+1,I)
144     Y(1,I)=Y(KSTEP+1,I)
145   *****
146   C* MAINTAIN SUCCESSFUL H CONSTANTLY FOR 3 TIMES BEFORE DOUBLING
147   C* *****
148   ISSTEP=ISSTEP+1
149   IF(ISSTEP .LT. 3) GO TO 49
150   ISSTEP=0
151   H=2.*H
152   IF(H .GT. HMAX) H=HMAX
153   GO TO 49
154 END
```

```
SUBROUTINE START(KSTEP,H,N,Y,YN,YOLD,INVERT,IT,METHOD,IAT)
C*****C
C* A SELF-STARTER, CALLED BY DIFEQ
C* ARGUMENTS ALREADY DEFINED IN MAIN PROGRAM
C* FINAL VALUES ARE STORED IN Y(J,I)
C* THIS PROGRAM CALLS 2 SUBROUTINES
C*      LMS(1.....)
C*      NLMS(1,...,IAT)
C*****C
```

```

15 PARAMETER NK=20, NC=20
16 IMPLICIT REAL*8 (A-H,O-Z)
17 EXTERNAL INVERT
18 COMMON A(NK,NC),ALPHA(4),T(NC)
19 DIMENSION Y(4,NC),YN(N),YOLD(4,NC)
20 AL=ALPHA(1)
21 ALPHA(1)=1.0
22 LO 1 I=1,N
23 1 YOLU(I,I)=Y(I,I)
24 UC 10 L=1,KSTEP
25 IF (METHOD .EQ. 1) GO TO 4
C***** FIRST-ORDER ADAMS-BASHFORTH METHOD TO START *****
C***** CALL NLMS(1,H,YOLD,N,YN,C,1,INVERT,IT,IAT)
C***** HA=H/4.
26 UC 3 I=1,4
27 UC 2 L=1,N
28 2 YOLU(I,L)=YN(L)
29 3 CONTINUE
30 GO TO 6
C***** GENERALIZE-ADAMS-BASHFORTH METHOD TO START *****
C***** CALL NLMS(1,H,YOLD,N,YN,C,1,INVERT,IT,IAT)
31 6 LO 5 I=1,N
32 Y(J+1,I)=YN(I)
33 5 YOLD(1,I)=YN(I)
34 10 CONTINUE
35 ALPHA(1)=AL
36 RETURN
37 ENC
38
39
40

```

```

1      SUBROUTINE NLMS(KSTEP,H,Y,N,YN,INDEX,IS,INVERT,IT,IAT)
2
3      C***NONLINEAR MULTISTEP ALGORITHM(NLMS), CALLED BY DIFEQ OR START
4      C***ARGUMENTS ALREADY DEFINED IN MAIN PROGRAM
5      C***THIS PROGRAM CALLS 3 SUBROUTINES
6      C***      INVERT(AH,...,P1)
7      C***      GfN(G,...,NC)
8      C***      PADE(A,...,INVERT)
9      C***      SOLUTION VECTOR IS STORED IN YN(I)
10     C***      PARAMETER NR=20, NC=20
11     C***      IMPLICIT REAL*8 (A-H,O-Z)
12
13     EXTERNAL INVERT
14     COMMON A(NR,NC),ALPHA(4),T(NC)
15     DIMENSION AH(NR,NC),AH2(NR,NC),AH3(NR,NC),AM4(NR,NC)
16     DIMENSION EAH(NR,NC),E2AH(NR,NC),E3AH(NR,NC),G(NC)
17     DIMENSION P(NR,NC),P1(NR,NC),PHI(4,NR,NC),QH1(4,NR,NC)
18     DIMENSION UNIT(NR,NC),Y(4,4,NR,NC)
19     DIMENSION AT(NR,NC),Y(4,NC),YN(NC)
20
21     C*      I5 IS AN INDICATOR WHEN IT IS 1, PROGRAM DOES INITIALIZATION,
22     C*      CALCULATES AND SAVES AH, EXP(AH), A INVERSE, PHI FUNCTION
23     C*      I5.GT.1 ABOVE CALCULATIONS ARE BYPASSED
24
25     IF(IS .GT. 1) GO TO (100,200,300), KSTEP
26     DO 1 I=1,N
27     DO 2 J=1,N
28     P1(I,J)=0.0
29     UNIT(I,J)=0.0
30     EAH(I,J)=0.0
31     E2AH(I,J)=0.0
32     E3AH(I,J)=0.0
33     AH2(I,J)=0.0
34     AH3(I,J)=0.0
35     2 AH4(I,J)=0.0
36     1 UNIT(I,J)=1.0

```

```

37      UO 3 I=1,N
38      UO 3 J=1,N
39      UO 3 K=1,N
40      PHI(1,J,K)=0.0
41      3 WHI(1,J,K)=0.0
42      UO 6 I=1,N
43      UO 6 J=1,N
44      6 AH(I,J)=H*A(I,J)
45      GO TO (100,200,300), KSTEP
46      100 CONTINUE
47      C*****NONLINEAR MULTI-1-STEP STARTS HERE. BEGINNING SECTION DOES
48      C*INITIALIZATION
49      C*****C
50      C*****C
51      LO 132 I=1,N
52      LO 133 J=1,N
53      P1(I,J)=0.0
54      YN(I)=0.0
55      PHI(2,I,1)=0.0
56      PHI(3,I,1)=0.0
57      IF (IS.GT.1 .AND. INDEX.EQ.0) GO TO 131
58      IF (IS.GT.1 .AND. INDEX.EG.1) GO TO 170
59      C*****C
60      R1 CONTAINS (AH)*(-1), EAh CONTAINS EXP(AH)
61      C*****C
62      IF (N-1) 120, 120
63      R1(1,1)=1./AH(1,1)
64      GO TO 142
65      120 CALL INVERT(AH,N,P1)
66      122 IF (N-1) 123, 123
67      EAh(1,1)=DEXP(A(1,1)*H)
68      GO TO 125
69      123 CALL PADE(A,H,EAh,E2AH,E3AH,HN,HN,NC,INVERT)
70      125 LO 103 I=1,N
71      LO 103 U=1,N
72      103 AH(I,U)=ALPHA(1)*EAH(I,U)+UNIT(I,U)

```

```

*****
C* EXPLICIT NLM-1-STEP METHODS
C* DO LOOP 105 CALCULATES PHI(1,0)
C* LOOP 106 OR 110 COMPUTES FINAL Y(N+1)
C* *****
74 IF(INDEX .NE. 0) GO TO 150
75 IF(IT .EQ. 0) GO TO 169
76 DO 105 I=1,N
77 DO 105 J=1,N
78 DO 105 K=1,N
79 105 PHI(1,I,J)=PHI(1,I,J)-P1(I,K)*AH(K,J)
80 131 CALL GFN(G,H,N,Y,KSTEP,T,A,NR,NC)
81 108 YN(I)=Y(I)-ALPHA(I)*EAH(I,J)*Y(I,J)+H*PHI(I,J)*G(J)
82 108 YN(I)=YN(I)-ALPHA(I)*EAH(I,J)*Y(I,J)+H*PHI(I,J)*G(J)
83 109 DO 110 I=1,N
84 110 YN(I)=YN(I)-ALPHA(I)*EAH(I,J)*Y(I,J)
85 110 RETURN
86 110 YN(I)=YN(I)-ALPHA(I)*EAH(I,J)*Y(I,J)
87 150 IF(N-1) 152,152
88 AH2(1,1)=P1(1,1)*P1(1,1)
89 152 DO 153 I=1,N
90 152 DO 154 J=1,N
91 152 DO 154 K=1,N
92 152 154 AH2(I,J)=AH2(I,J)+P1(I,K)*P1(K,J)
93 154 AH2(I,J)=AH2(I,J)+P1(I,K)*P1(K,J)
94 153 IF(IT .EQ. 0) GO TO 167
95 153 DO 160 I=1,N
96 160 DO 161 J=1,N
97 160 DO 162 K=1,N
98 162 PHI(1,I,J)=PHI(1,I,J)+ALPHA(I)*H*A(I,K)*EAH(K,J)
99 PHI(1,I,J)=PHI(1,I,J)-AH(I,J)
100

```

```

111      E2AH(I,J)=AH(I,J)+H*A(I,J)
112      CONTINUE
113      K2=KSTEP+1
114      IF(IT .EQ. 0) GO TO 167
115      DO 163 I=1,N
116      DO 164 K=1,K2
117      CALL GFN(G,H,N,Y,K,T,A,NR,NC)
118      DO 165 J=1,N
119      IF(K .EQ. 1) PHI(3,I,1)=PHI(3,I,1)+PHI(1,I,J)*G(J)
120      IF(K .EQ. 2) PHI(2,I,1)=PHI(2,I,1)+E2AH(I,J)*G(J)
121      PHI(3,I,1)=PHI(3,I,1)+PHI(2,I,1)
122      163 CONTINUE
123      DO 166 I=1,N
124      DO 166 K=1,N
125      166 YN(I)=YN(I)-H*AH2(I,K)*PHI(3,K,1)-ALPHA(I)*EAH(I,K)*Y(I,K)
126      DO 167 I=1,N
127      167 DO 168 K=1,N
128      168 YJ(I)=YN(I)-ALPHA(I)*EAH(I,K)*Y(I,K)
129      ****
130      C* IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION,
131      C* EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1)
132      ****
133      ****
134      172 IF(IAT .EQ. 0) RETURN
135      176 TH=T(1)+H
136      CALL AFNT(AT,N,TH,NR,NC)
137      DO 173 I=1,N
138      U(I)=0.0
139      DO 173 J=1,N
140      G(I)=G(I)+(AT(I,J)-A(I,J))*Y(2,J)
141      DO 174 I=1,N
142      GH1(4,I)=0.0
143      DO 174 J=1,N
144      GH1(4,4,I)=GH1(4,4,I)+E2AH(I,J)*G(J)
145      DO 175 I=1,N
146      DO 175 J=1,N

```

```

147      YN(I)=YN(I)-H*AH2(I,J)*QHI(4,4,J)
148      RETURN
149      200 CONTINUE
150      C*****NONLINEAR MULTI-2-STEP STARTS HERE. BEGINNING SECTION DOES
151      C* INITIALIZATION
152      C*****INITIALIZATION
153      UO 240 I=1,N
154      UO 241 J=1,N
155      241 P1(I,J)=0.0
156      YN(I)=0.0
157
158      240 PH1(4,1,1)=0.0
159      IF (IS .GT. 1) GO TO 212
160      C*****EAH CONTAINS EXP(AH), E2AH CONTAINS EXP(2AH)
161      C*****EAH(N-1)=DEXP(AH(1,1))
162      IF (N-1) 208, 208
163      EAH(1,1)=DEXP(AH(1,1))
164      E2AH(1,1)=EAH(1,1)*EAH(1,1)
165      GO TO 212
166      208 CALL PADE(A,H,EAH,E2AH,E3AH,G,N,NR,NC,INVERT)
167      H2=H+H
168      CALL PADE(A,H2,E2AH,E3AH,AH4,G,N,NR,NC,INVERT)
169      212 IF (INDEX .GT. 0) GO TO 260
170
171      C*****EXPLICIT NLH-2-STEP METHODS
172      C* AH2 CONTAINS (AH)**(-2)
173      C*****AH2 CONTAINS (AH)**(-2)
174      C*****AH2 CONTAINS (AH)**(-2)
175      IF (IS .GT. 1) GO TO 234
176      251 IF (N-1) 201, 201
177      AH2(1,1)=1./AH(1,1)**2
178      GO TO 206
179      201 CALL INVERT(AH,N,P1)
180      UO 203 I=1,N
181      UO 203 J=1,N
182      UO 203 K=1,N

```

```

183      . 203 AH2(I,J)=AH2(I,J)+P1(I,K)*P1(K,J)
184      .IF (IT .EQ. 0) GO TO 235
185
186      C*   AI THE FINISH OF LOOP 215 --
187      C*   PHI(I,I,J) CONTAINS PHI(1,0), PHI(2,I,J) CONTAINS PHI(2,0)
188      C*   FINAL Y(N+1) ARE CALCULATED IN LOOP 232
189
190      206 U0 213 I=1,N
191      DO 213 J=1,N
192      213 P1(I,J)=ALPHA(1)*E2AH(I,J)+ALPHA(2)*EAH(I,J)+UNIT(I,J)
193      U0 215 I=1,N
194      U0 215 J=1,N
195      U0 215 K=1,N
196      PHI(I,1,J)=PHI(I,1,J)+ALPHA(1)*AH(I,K)*E2AH(K,J)
197      PHI(I,2,J)=PHI(I,2,J)+ALPHA(2)*AH(I,K)*EAH(K,J)
198      U0 216 I=1,N
199      U0 219 J=1,N
200      PHI(I,1,J)=PHI(I,1,J)-P1(I,J)-AH(I,J)
201      PHI(I,2,J)=PHI(I,2,J)+P1(I,J)+2.*AH(I,J)
202      P1(I,I)=0.
203      U0 220 K=1,KSTEP
204      CALL GFN(G,H,N,Y,K,T,A,MK,NC)
205      U0 221 I=1,N
206      U0 221 J=1,N
207      221 YN(I)=YN(I)+PHI(K,I,J)*G(J)*H
208      220 CONTINUE
209      U0 230 I=1,N
210      U0 230 J=1,N
211      230 P1(I,I)=P1(I,I)-AH2(I,J)*YN(J)
212      235 U0 232 I=1,N
213      IF (IT .EQ. 0) P1(I,I)=0.0
214      U0 233 J=1,N
215      233 PHI(4,I,1)=PHI(4,I,1)-ALPHA(2)*EAH(I,J)*Y(2,J)-ALPHA(1)*E2AH(I,J)*
216      SY(1,J)
217      232 IN(I)=P1(I,I)+PHI(4,I,1)
218      IF (IAT .EQ. 0) RETURN

```

TR 5341

```
219      DO 290 I=1,N
220      DO 290 J=1,N
221      290 E2AH(I,J)=PHI(2,I,J)
222      GO TO 176
223      *****
224      C* IMPLICIT NONLINEAR MULTI-2-STEP METHODS
225      C* NEXT DELTA 267, AH3 CONTAINS (AH)**(-3)
226      C*****
227      260 CONTINUE
228      IF (IS .GT. 1) GO TO 284
229      IF(N-1) 262, 262
230      AH2(1,1)=AH(1,1)*AH(1,1)
231      AH3(1,1)=1./(AH2(1,1)*AH(1,1))
232      GO TO 263
233      262  DO 264 I=1,N
234      DO 264 J=1,N
235      DO 264 K=1,N
236      AH2(I,J)=AH2(I,J)+AH(I,K)*AH(K,J)
237      DO 267 I=1,N
238      DO 267 J=1,N
239      AH4(I,J)=0.0
240      DO 267 K=1,N
241      AH4(I,J)=AH4(I,J)+AH2(I,K)*AH(K,J)
242      CALL INVERT(AH4,N,AH3)
243      263 CONTINUE
244      284 IF (IS .GT. 1) GO TO 279
245      IF (IT .EQ. 0) GO TO 285
246      *****
247      C* END OF LOOP 275, PHI(L,I,J) CONTAIN PHI(2,L), L=0,1,2
248      C* FINAL Y(N+1) ARE CALCULATED IN LOOP 282 OR 286
249      *****
250      DO 270 I=1,N
251      DO 270 J=1,N
252      *(1,1,1,J)=UNIT(I,J)-1.5*AH(I,J)+AH2(I,J)
253      *(1,2,1,J)=UNIT(I,J)-0.5*AH(I,J)
254      *(1,3,1,J)=UNIT(I,J)+0.5*AH(I,J)
```

```

* (2,1,I,J)=-2.*UNIT(I,J)-AH(I,J)
* (2,2,I,J)=-2.*UNIT(I,J)+AH2(I,J)
* (2,3,I,J)=-2.*UNIT(I,J)+AH(I,J)
* (3,1,I,J)=W(1,2,I,J)
* (3,2,I,J)=W(1,3,I,J)
* (3,3,I,J)=UNIT(I,J)+1.5*AH(I,J)+AH2(I,J)
270 W(3,3,I,J)=UNIT(I,J)+1.5*AH(I,J)+AH2(I,J)

261 DO 283 K=1,3
262 DO 272 I=1,N
263 DO 273 J=1,N
264 DO 274 L=1,N
265 GHI(K,I,J)=QHI(K,I,J)+ALPHA(1)*W(K,1,I,L)*E2AH(L,J)+ALPHA(2)*W(K,2
266 S,I,L)*EAH(L,J)
267 GHI(K,I,J)=GHI(K,I,J)+W(K,3,I,J)
268 <72 CONTINUE
269 283 CONTINUE
270 DO 275 L=1,3
271 DO 275 I=1,N
272 DO 275 J=1,N
273 DO 275 K=1,N
274 PHI(L,I,J)=PHI(L,I,J)-AH3(I,K)*QHI(L,K,J)
275 DO 280 I=1,N
276 DO 280 K=1,3
277 CALL GFN(G,H,N,Y,K,T,A,N,K,NC)
278 DO 282 J=1,N
279 YN(I)=YN(I)+PHI(K,I,J)*G(J)**H
280 IF(K.EQ.3) YN(I)=YN(I)-ALPHA(1)*E2AH(I,J)*Y(1,J)-ALPHA(2)*EAH(I,
281 J)*Y(2,J)
280 CONTINUE
282 DO 293 I=1,N
283 DO 286 I=1,N
284 DO 286 J=1,N
285 DO 286 K=1,N
286 YN(I)=YN(I)-ALPHA(1)*E2AH(I,J)*Y(1,J)-ALPHA(2)*EAH(I,J)*Y(2,J)
287 293 IF(IAT.EQ.0) RETURN
C*****IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION,
C* EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1),
C*****A T(I)) AND ADJUST FINAL Y(N+1),
C*****

```

```

297 T1=T(1)+H
298 T2=T1+H
299 CALL AFNT(P,N,T1,MR,NC)
300 CALL AFNT(P,T,N,T2,MR,NC)
301 DO 294 I=1,N
302 G(1)=0.0
303 P1(1,I)=0.0
304 DO 294 J=1,N
305 G(I)=G(I)+(P(I,J)-A(I,J))*Y(2,J)
306 P1(1,I)=P1(1,I)+(AT(I,J)-A(I,J))*Y(3,J)
307 DO 295 I=1,N
308 P1(2,I)=0.0
309 DO 295 J=1,N
310 P1(2,I)=P1(2,I)+PHI(2,I,J)*G(J)+PHI(3,I,J)*P1(1,J)
311 IF(KSTEP.EQ. 3) GO TO 298
312 RETURN
313 296 YN(I)=YN(I)+H*P1(2,I)
314 297 YN(I)=YN(I)-H*P1(2,I)
315 298 DO 299 I=1,N
316 299 YN(I)=YN(I)-H*P1(2,I)
317 300 RETURN
318 ***** CONTINUE *****
319 C***** NONLINEAR MULTI-3-STEP STARTS HERE. BEGINNING SECTION DOES
320 C***** INITIAL CALCULATIONS
321 C***** THE FOLLOWING RESULTS ARE STORED ***
322 C***** BEFORE 303. THE FOLLOWING RESULTS ARE STORED ***
323 C***** EAH--EXP(AH), E2AH--EXP(2AH), E3AH--EXP(3AH), AM3--(AH)**(-3)
324 C***** EAH--EXP(AH), E2AH--EXP(2AH), E3AH--EXP(3AH), AM3--(AH)**(-3)
325 C***** KUP=KSTEP
326 C***** KUP=INDEX .EQ. 1) KUP=KSTEP+1
327 C***** IF(INDEX =0.0
328 C***** 1F(IIS .GT. 1) GO TO 321
329 C***** 1F(N-1) 302 . 302
330 C***** EAH(I,J)=DEXP((A(I,J)*H)
331 C***** E2AH(I,J)=EAH(I,J)*EAH(I,J)
332 C***** E3AH(I,J)=EAH(I,J)*EAH(I,J)*EAH(I,J)
333

```

```

328 E3AH(1,1)=E2AH(1,1)*EAH(1,1)
329 AH2(1,1)=AH(1,1)*AH(1,1)
330 IF(INDEX.EQ. 1) GO TO 350
331 AH3(1,1)=1./((AH2(1,1)**AH(1,1)))
332 GO TO 303
302 DO 330 I=1,N
333 DO 330 J=1,N
334 DO 330 K=1,N
335 DO 330 AH2(I,J)=AH2(I,J)+AH(I,K)*AH(K,J)
336 DO 333 I=1,N
337 DO 333 J=1,N
338 DO 333 K=1,N
339 AH4(I,J)=0.0
340 AH4(I,J)=AH4(I,J)+AH2(I,K)*AH(K,J)
341 IF(INDEX.EQ. 1) GO TO 351
342 CALL INVERT(AH4,N,AH3)
343 CALL PADE(A,H,EAH,E2AH,E3AH,G,N,NR,NC,INVERT)
344 H2=H+H
345 CALL PADE(A,H2,E2AH,E3AH,AH4,G,N,NR,NC,INVERT)
346 H3=H2+H
347 CALL PADE(A,H3,E3AH,AH4,W(1,1,1,1),G,N,NR,NC,INVERT)
348 303 IF(IT.EQ. 0) GO TO 370
349 C*****
350 C*****
C* CALCULATE PHI(3,K), K=0,1,2. RESULTS IN PHI(K,I,J)
C***** C*****
351 352
353 DO 304 I=1,N
354 DO 304 J=1,N
355 W(1,1,1,J)=UNIT(1,J)-1.5*AH(I,J)+AH2(I,J)
356 W(1,2,1,J)=UNIT(1,J)-0.5*AH(I,J)
357 W(1,3,1,J)=UNIT(1,J)+0.5*AH(I,J)
358 W(1,4,1,J)=UNIT(1,J)+1.5*AH(I,J)+AH2(I,J)
359 W(2,1,1,J)=2.*UNIT(I,J)-AH(I,J)
360 W(2,2,1,J)=-2.*UNIT(I,J)+AH2(I,J)
361 W(2,3,1,J)=-2.*UNIT(I,J)+AH(I,J)
362 W(2,4,1,J)=-2.*UNIT(I,J)-4.*AH(I,J)-3.*AH2(I,J)
363

```

TR 5341

```
      "(3,2,I,J)=UNIT(I,J)+0.5*AH(I,J)
      "(3,3,I,J)=W(1,4,I,J)
304      W(3,4,I,J)=UNIT(I,J)+2.5*AH(I,J)+3.*AH2(I,J)
360      UO 306 I=1,KUP
            UO 307 I=1,N
            DC 308 J=1,N
            UO 309 K=1,N
309      EH1(II,I,J)=QHI(II,I,J)+ALPHA(1)*W(II,1,I,K)*E3AH(K,J)+ALPHA(2)*W(
311,2,I,K)*E2AH(K,J)+ALPHA(3)*W(II,3,I,K)*EAH(K,J)
308      QHI(II,I,J)=QHI(II,I,J)+W(II,4,I,J)
307      CONTINUE
306      CONTINUE
370      EH1(II,I,J)=QHI(II,I,J)+ALPHA(1)*W(II,1,I,K)*E3AH(K,J)+ALPHA(2)*W(
311,2,I,K)*E2AH(K,J)+ALPHA(3)*W(II,3,I,K)*EAH(K,J)
373      QHI(II,I,J)=QHI(II,I,J)+W(II,4,I,J)
374      CONTINUE
375      CONTINUE
376      UO 310 K=1,KUP
377      UO 310 I=1,N
378      UO 310 J=1,N
379      UO 310 L=1,N
380      IF(INDEX.EQ.0) PHI(K,I,J)=2*U(1)(K,I,J)-AH3(I,L)*GHI(K,L,J)
310      IF(INDEX.EQ.1) PHI(K,I,J)=PHI(K,I,J)-AH4(I,L)*GHI(K,L,J)
321      DO 314 K=1,KUP
            CALL GFN(G,H,N,Y,K,T,A,NR,NC)
            UO 315 I=1,N
            UO 316 J=1,N
C*****C*****C*****C*****C*****C*****C*****C*****C*****C*****
C*   CALCULATE FINAL Y(N+1)
C*****C*****C*****C*****C*****C*****C*****C*****C*****C*****
388      YN(I)=YN(I)+H*PHI(K,I,J)*G(J)
389      IF(K.EQ. KUP) YN(I)=YN(I)-ALPHA(3)*EAH(I,J)*Y(3,J)-ALPHA(2)*E2AH(
311,J)*Y(2,J)-ALPHA(1)*E3AH(I,J)*Y(1,J)
390      CONTINUE
314      CONTINUE
391      GO TO 340
370      UO 371 I=1,N
            UO 371 J=1,N
371      YN(I)=YN(I)-ALPHA(3)*EAH(I,J)*Y(3,J)-ALPHA(2)*E2AH(I,J)*Y(2,J)-ALP
1MA(1)*E3AH(I,J)*Y(1,J)
```

TR 5341

```

394 340 IF(IAT.EQ.0) RETURN
400 401 IF(INULA.EQ.0) GO TO 297
402 C* T1 SEE IS A FUNCTION OF T
403 C* *****
404 11=T(1)+H
405 T2=T1+H
406 T3=T2+h
407 CALL AFNT(AT,N,T1,NR,NC)
408 CALL AFNT(P,N,T2,NR,NC)
409 CALL AFNT(P1,N,T3,NR,NC)
410 UO 341 I=1,N
411 G(1)=0.0
412 GHI(4,3,1)=0.0
413 GHI(4,4,1)=C.0
414 UO 341 J=1,N
415 G(1)=G(1)+(AT(I,J)-A(I,J))*Y(2,J)
416 GHI(4,3,1)=GHI(4,3,I)+(P(I,J)-A(I,J))*Y(3,J)
417 GHI(4,4,1)=GHI(4,4,I)+(P1(I,J)-A(I,J))*Y(4,J)
418 UO 342 I=1,N
419 UO 342 J=1,N
420 342 YN(I)=YN(I)+H*(PHI(2,I,J)*G(J)+PHI(4,I,J)*QH
421 S1(4,4,J))
422 RETURN
423 C* *****
424 C* CALCULATE IMPLICIT PHI FUNCTION, PHI(3,J), J=0,1,2,3
425 C* *****
426 350 AH3(1,1)=AH2(1,1)*AH(1,1)
427 AH4(1,1)=1./((AH3(1,1)*AH(1,1))
428 GO TO 357
429 UO 351 J=4 I=1,N
430 UO 354 J=1,N
431 EAH(1,J)=0.0
432 AH3(1,J)=AH4(1,J)
433 UO 355 K=1,N

```

TR 5341

```
434 EAH(I,J)=EAH(I,J)+AH4(I,K)*AH(K,J)
435 CALL INVERT(EAH,N,AH4)
436 CALL PAUE(A,H,EAH,E2AH,E3AH,G,N,NR,NC,INVERT)
437 H2=H+H
438 CALL PAUE(A,H2,E2AH,E3AH,W(1,1,1,1),G,N,NR,NC,INVERT)
439 H3=H2+H
440 CALL PAUE(A,H3,E3AH,W(1,1,1,1),G,N,NR,NC,INVERT)
441 IF(IT.EQ.0) GO TO 370
442 DO 358 I=1,N
443 LO 358 J=1,N
444 *(1,1,I,J)=-UNIT(I,J)+2.*AH(I,J)-11.*AH2(I,J)/6.+AH3(I,J)
445 *(1,2,I,J)=-UNIT(I,J)+AH(I,J)-AH2(I,J)/3.
446 *(1,3,I,J)=-UNIT(I,J)+AH2(I,J)/6.
447 *(1,4,I,J)=-UNIT(I,J)-AH(I,J)+AH2(I,J)/3.
448 *(2,1,I,J)=3.*UNIT(I,J)-5.*AH(I,J)+3.*AH2(I,J)
449 *(2,2,I,J)=3.*UNIT(I,J)-2.*AH(I,J)-0.5*AH2(I,J)+AH3(I,J)
450 *(2,3,I,J)=3.*UNIT(I,J)+AH(I,J)-AH2(I,J)
451 *(2,4,I,J)=3.*UNIT(I,J)+4.*AH(I,J)+1.5*AH2(I,J)
452 *(3,1,I,J)=-3.*UNIT(I,J)+4.*AH(I,J)-1.5*AH2(I,J)
453 *(3,2,I,J)=-3.*UNIT(I,J)+AH(I,J)+AH2(I,J)
454 *(3,3,I,J)=-3.*UNIT(I,J)-2.*AH(I,J)+0.5*AH2(I,J)+AH3(I,J)
455 *(3,4,I,J)=-3.*UNIT(I,J)-5.*AH(I,J)-3.*AH2(I,J)
456 *(4,1,I,J)=-W(1,2,I,J)
457 *(4,2,I,J)=-W(1,3,I,J)
458 *(4,3,I,J)=-W(1,4,I,J)
459 *(4,4,I,J)=UNIT(I,J)+2.*AH(I,J)+11.*AH2(I,J)/6.+AH3(I,J)
460 GO TO 360
461 END
```

```
1 SUBROUTINE LMS(KSTEP,H,Y,N,YN,INDEX)
2 ****
3 C* LINEAR MULTISTEP METHODS (STEP NUMBER.LE.3)
* ****
```

```

4 C*   BETA COEFFICIENTS ARE FORMULATED TO DEPEND UPON
5   THE CHARACTERISTIC COEFFICIENTS, ALPHA
C*   100 THRU 400 CALCULATES BETA. CALCULATIONS ARE NEEDED ONLY ONCE
C*   800 BETA'S OF EXPLICIT METHODS ARE STORED IN B1. OF IMPLICIT, IN B2
C*   THIS SUBROUTINE IS CALLED BY START OR DIFEQ
C***** ****
10 PARAMETER NR=20, NC=20
11 IMPLICIT REAL*8 (A-H,O-Z)
12 COMMON A(NR,NC),ALPHA(4),T(NK)
13 DIMENSION Y(4,NC),YN(N),B(4),B1(3),B2(4),FN(NC)
14 DATA IBETA/0/
15 KSTEP
16 DO 1 I=1,N
17 1 YN(I)=0.0
18 IF (IBETA .GT. 0) GO TO 410
19 GO TO (100,200,300),KSTEP
20 100 B1(1)=1.0
21 B2(1)=0.
22 B2(2)=B2(1)
23 GO TO 400
24 200 B1(1)=0.5*ALPHA(2)
25 B1(2)=B1(1)+2.
26 B2(1)=5.*ALPHA(2)/12.+1./3.
27 B2(2)=2.*ALPHA(2)/3.+4./3.
28 B2(3)=-5.*ALPHA(2)/12.+1./3.
29 GO TO 400
30 300 B1(1)=5.*ALPHA(2)/12.+ALPHA(3)/3.+0.75
31 B1(2)=2.*ALPHA(2)/3.+4.*ALPHA(3)/3.
32 B1(3)=-ALPHA(2)/12.+ALPHA(3)/3.+2.25
33 B2(1)=3.*ALPHA(2)/8.+ALPHA(3)/3.+3./8.
34 B2(2)=19.*ALPHA(2)/24.+4.*ALPHA(3)/3.+9./8.
35 B2(3)=-5.*ALPHA(2)/24.+ALPHA(3)/3.+9./8.
36 B2(4)=ALPHA(2)/24.+3./8.
37 400 BETA=1
38 1F(KSTEP .EQ. 1) IUEIA=0
39 410 1F(INDEX .GT. 0) GO TO 500

```

```

        UO 415 I=1,3
415 B(I)=B1(I)
420 DO 450 I=1,KSTEP
421      CALL FFN(Y,N,FN,I)
422      UO 440 J=1,N
423          YN(J)=YN(J)+H*B(I)*FN(J)
424
425      CCNTINUE
426
427      IF(K .EQ. 0) GO TO 465
428      CALL FFN(Y,N,FN,K)
429      UO 430 J=1,N
430          YN(J)=YN(J)+H*B(K)*FN(J)
431
432      UO 470 I=1,N
433          UO 470 J=1,KSTEP
434              YN(I)=YN(I)-ALPHA(I)*Y(J+I)
435
436      CCNTINUE
437      RETURN
438
439      K=KSTEP+1
440      DO 510 I=1,4
441          D(I)=D2(I)
442      GO TO 420
443
444      END
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461

```

```

***** SUBROUTINE PADE(A,H,P,B,C,COL,N,NR,NC,INVERT) *****
C***** CALCULATE MATRIX EXPONENTIAL BY PADE APPROXIMATION *****
C***** CALLED BY NLMS SUBROUTINE *****
C***** CALLS FOR SUBROUTINE INVERT *****
C***** IMPLICIT REAL*8 (A-H,O-Z) *****
C***** DIMENSION A(NR,NC),P(NR,NC),B(NR,NC),C(NR,NC),COL(NC) *****
C***** EXTERNAL INVERT *****
C***** DATA BETA/.30600/ *****
C***** HAVE=H *****
C***** UO 2 I=1,N *****
C***** UO 1 J=1,N *****
1 2 3 4 5 6 7 8 9 10 11 12 13

```

```

14      B(I,J)=0.0
15      C(I,J)=0.0
16      P(I,J)=0.0
17      1 CONTINUE
18      COL(I)=0.0
19      2 CONTINUE
20      DO 17 I=1,N
21      DO 16 J=1,N
22      CCL(I)=COL(I)+DABS(A(J,I))
23      16 CONTINUE
24      17 CONTINUE
25      XNORM=COL(1)
26      DO 18 I=1,N
27      IF (XNORM .GT. COL(I)) GO TO 18
28      XNORM=COL(I)
29      18 CONTINUE
30      C*****H=0.0E(.1) 3,20,20
31      C* COLUMN NORM IS USED TO SEE WHETHER EXP(A) NEEDS REDUCTION
32      C*****H=0
33
34      30 IF (XNORM*H-0.0E(.1)) 3,20,20
35      C*****H=0
36      C* EXP(A)=(I-(.5+BETA)*A+BETA*A**2)**(1+(.5-BETA)*A)
37      C*****H=0
38      3 DO 6 I=1,N
39      DO 5 J=1,N
40      DO 4 K=1,N
41      P(I,J)=P(I,J)+A(I,K)*A(K,J)
42      4 CONTINUE
43      C(I,J)=(BETA*A(P(I,J))*H-.5D0+BETA)*A(I,J))*H
44      5 CONTINUE
45      C(I,I)=C(I,I)+1.0
46      6 CONTINUE
47      CALL INVERT(C,N,B)
48      DO 9 I=1,N
49      DO 10 J=1,N

```

```

50
51      C(I,J)=(.5D0-BETA)*A(I,J)*H
52      P(I,J)=0.0
53      10 CONTINUE
54          C(I,I)=C(I,I)+1.0
55          9 CONTINUE
56          DO 12 I=1,N
57              DO 13 J=1,N
58                  DO 14 K=1,N
59                      P(I,J)=P(I,J)+B(I,K)*C(K,J)
60
61          14 CONTINUE
62          13 CONTINUE
63          12 CONTINUE
64          IF(M .EQ. 0) 60 TO 40
65          C***NORM(AH).GT.(.1), EXP(AH)=EXP(A/2k*M)*(2*k*M)
66          DO 24 I=1,N
67              DO 25 J=1,N
68                  B(I,J)=0.0
69          25 CONTINUE
70          24 CONTINUE
71          DO 36 K=1,M
72              DO 27 I=1,N
73                  DO 28 J=1,N
74                  DO 29 L=1,N
75                      B(I,J)=B(I,J)+P(I,L)*P(L,J)
76          29 CONTINUE
77          28 CONTINUE
78          27 CONTINUE
79          DO 31 I=1,N
80              DO 32 J=1,N
81                  P(I,J)=B(I,J)
82                  B(I,J)=0.0
83          32 CONTINUE
84          31 CONTINUE
85

```

```

86      H=HAVE
87      RETURN
88      H=H/2.0
89      M=M+1
90      DO 54 I=1,N
91      DO 55 J=1,N
92      P(I,J)=0.0
93      CONTINUE
94      CONTINUE
95      GO TO 30
96      H=HAVE
97      RETURN
98      END

```

```

SUBROUTINE INVERT(A,N,ANS)
C*****C*****C*****C*****C*****C*****C*****C*****C*****C*****C*****C*****
C* MATRIX INVERSION SUBROUTINE, CALLED BY PAGE OR NLMS
C* A CONTAINS THE ORIGINAL ELEMENTS AND REMAINS UNALTERED
C* ANS CONTAINS THE A*(-1)
C* THIS SETUP IS USING UNIVAC 1108 MATHPK EXISTING LCUBLE
C* PRECISION GAUSS-JORDAN REDUCTION
C* THIS PROGRAM IS REPLACEABLE BY THE USER
C*****C*****C*****C*****C*****C*****C*****C*****C*****C*****C*****C*****
PARAMETER NR=20, NC=20
10  PLICIT REAL*8 (A-H,O-Z)
11  DIMENSION A(NK,NK),ANS(NK,NC)
12  DIMENSION V(2),JC(NC)
13  V(1)=1.0
14  JC(1)=1.0
15  DO 1 I=1,N
16  DO 1 J=1,N
17  1 ANS(I,J)=A(I,J)
18  CALL DGUK(ANS,NK,NC,N,NC,0,V)
19  RETURN
20  WRITE(4,2)
21  FORMAT(3X,'MATRIX INVERSION ERROR')
22  RETURN
23  END

```

```

1  SUBROUTINE GFN(G,H,N,Y,J,T,A,NR,NC)
2    IMPLICIT REAL*8 (A-H,O-Z)
3    DIMENSION A(NR,NC),Y(4,NC),G(NC),T(NC)
4    *****
5    C* CALCULATES THE G(T,Y). ALL CALCULATIONS TO BE INSERTED HERE
6    C* CALLED BY NLMS OR DIFEQ
7    C* E.G.   Y/DT=-100Y+(1+T**2)
8    C* DEFINE G(I)=1.+T(J)*T(J)
9    C* *****
10   RETURN
11   END

```

```

1  SUBROUTINE FFN(Y,N,FN,I)
2    PARAMETER NR=20, NC=20
3    IMPLICIT REAL*8 (A-H,O-Z)
4    COMMON A(NR,NC),ALPHA(4),T(NC)
5    DIMENSION FN(I),Y(4,NC)
6    *****
7    C* CALCULATES THE F(T,Y). ALL CALCULATIONS ENTER HERE
8    C* CALLED BY SUBROUTINE LMS
9    C* E.G.   Y/DT=-100Y+(1+T**2)
10   C* DEFINE F(I)=-100.*Y(I,1)+(1.+T(I))*T(I)
11   C* *****
12   RETURN
13   END

```

```

1  SUBROUTINE AFNI(A,N,T,NR,NC)
2    DOUBLE PRECISION A(NR,NC),T
3    *****
4    C* MATRIX A IS A FUNCTION OF T, ENTER ALL COMPUTATIONS HERE
5    C* E.G.   A(T)=-100T
6    C* DEFINE A(I,1)=-100.*T
7    C* *****
8    RETURN
9    END

```

```

SUBROUTINE PRINT(H,T,Y,N,I)
DOUBLE PRECISION H,T,Y(I)
C***** DESIGNED FOR USER TO PRINTOUT INTERMEDIATE RESULTS, Y(T)
C*   USER CAN DEFINE HIS OWN PRINTOUT FORMATS
C***** 12 FORMAT(1X,2(E15.6,2X),4X,5(E15.8,2X))
1184 WRITE (4,12) H,T,(Y(J),J=1,N)
34 CONTINUE
      RETURN
      END

```

```

SUBROUTINE BEGIN(KSTEP,H,N,Y,YOLD,INVERT,IT,METHOD,IAT)
PARAMETER NR=20, NC=20
IMPLICIT REAL*8 (A-H,O-Z)
EXTERNAL INVERT
COMMON A(NR,NC),ALPHA(4),T(NC)
DIMENSION Y(4,NC),YN(N),YOLD(4,NC)
C***** DESIGNED FOR USER TO SUPPLY HIS OWN STARTER
C*   ALL CALCULATIONS ENTER HERE
C***** RETURN
      END

```

ANSI FORTRAN

TR 5341

```

1      C
2      C THIS PROGRAM COMMANDS A SET OF SUBROUTINES TO INTEGRATE A
3      C SYSTEM OF FIRST-ORDER ORDINARY DIFFERENTIAL EQUATIONS OF
4      C THE FORM
5          DY(T)/DTEAY + G(T,Y)
6          OR
7          DY(T)/DTEF(T,Y)

8      OVER THE CLOSED INTERVAL-(T(1),TMAX). THIS IS A
9      FIXED-ORDERING, VARIABLE(FIXED AS WELL)-STEP-SIZE PROGRAM.
10     ALL INPUTS MUST BE GIVEN. USERS WHO HAVE NO KNOWLEDGE TO
11     SELECT ALPHA FOR STABILITY, WE SUGGEST ---
12     KSTEP=1 -- ALPHA(1),(2) = -1.0D0, 1.0D0
13     KSTEP=2 -- ALPHA(1) . . . (3) = 0.0D0, -1.0D0, 1.0D0
14     KSTEP=3 -- ALPHA(1) . . . (4) = 0.0D0, 0.0D0, -1.0D0, 1.0D0
15     ONLY DIFEQ IS CALLED BY THIS PROGRAM. OTHER SUBROUTINES ARE
16     CALLED BY SUBROUTINES. UNUSED SUBROUTINES ARE DEFINED
17     DUMMIES TO SATISFY THE COMPILER. ALL COMPUTATIONS ARE
18     PERFORMED IN DOUBLE PRECISION. INPUT PARAMETERS HAVE THE
19     FOLLOWING DEFINITIONS
20
21     A      A 2-DIMENSIONAL ARRAY OF (N X N) FOR STORING
22     ELEMENTS OF MATRIX A
23
24     ALPHA CHARACTERISTIC POLYNOMIAL COEFFICIENTS. NO CONCERN
25     TO THE USER. ADAMS COEFFICIENTS ARE SUGGESTED
26     FOR STRONG STABILITY
27
28     ERR    USER-REQUIRED TOLERANCE
29
30     H      USER-SUGGESTED INITIAL STEP SIZE.
31
32     HMAX   THE MAXIMUM STEP SIZE THE USER ALLOWS THE PROGRAM
33     TO CONSIDER
34

```

```

35      IAT    SET =0, IF A IS A CONSTANT
36      SET =1, IF A IS A FUNCTION OF T
37      IGFN   SET =0, IF G(T,Y) IS ZERO
38      SET =1, IF G(T,Y) IS NONZERO
39      ****
40      INDEX  SET =0 FOR EXPLICIT
41          SET =1 FOR IMPLICIT
42      IPC    SET =0 FOR PREDICTOR OR CORRECTOR
43          SET =1 FOR PREDICTOR-AND-CORRECTOR
44      ****
45      KSTEP STEP iNUMBER (LESS THAN 4)
46      ****
47      METHOD SET =C, LS METHODS
48          SET =1, NLMS METHODS
49      ****
50      N      NUMBER OF EQUATIONS
51      ****
52      T      A 1-DIMENSIONAL ARRAY. T(1) CONTAINS INITIAL TIME
53      TMAX  FINAL TIME, ASSIGNED BY THE USER
54      YZERO A 1-DIMENSIONAL ARRAY, CONTAINING INITIAL Y
55      ****
56      READ FOLLOWING INPUTS ACCORDING TO USERS PREFERABLE
57      FORMATS AND HIS ORDER
58      N, KSTEP, IAT, IGH-N, INDEX, IPC, METHOD
59      H, RMAX, T(1), TMAX, ERR
60      (ALPHA(L+1), L=0, KSTEP)
61      (YZERO(K), K=1, N)
62      ((A(I,J), I=1, N), J=1, N)
63      ****
64      ****
65      ****
66      ****
67      ****
68      ****
69      ****
CUMULATIVE A(20,20), ALPHAS(4), T(20)
DIMENSION Y(4,20), YZERO(20)

```

```

70      DOUBLE PRECISION A,ALPHA,ERR,H,HMAX,T,TMAX,Y,YZERO
71      DATA ERR/.1D-11/,H/.1D-2/,HMAX/.1D0/,INDEX/0/,KSTEP/1/,METHOD/1/
72      C
73      C      * USER DEFINES HIS INPUTS HERE
74      C
75      C      10 READ INPUTS
76      CALL LIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,Y,YZERO,IGFN,IPC,METHOD,
77      INDEX,IAT)
78      C      GO TO 10
    END

1      SUBROUTINE START(KSTEP,H,N,Y,YOLD,IT,METHOD,IAT)
2      C
3      C      * A SELF-STARTER, CALLED BY DIFEQ
4      C      * ARGUMENTS ALREADY DEFINED IN MAIN
5      C      * FINAL VALUES ARE STORED IN Y(J,I)
6      C      * THIS PROGRAM CALLS 2 SUBROUTINES
7      LMS(1,0,0,0)
8      NLMS(1,0,0,IAT)
9
10     COMMON A(20,20),ALPHA(4),T(20)
11     DIMENSION Y(4,20),YN(20),YOLD(4,20)
12     DOUBLE PRECISION A,AL,ALPHA,H,HA,T,Y,YN,YOLD
13     DATA 10,11/0,1/
14     AL=ALPHA(1)
15     ALPHA(1)=-1.00
16     DO 1 I=1,N
17     YOLD(1,I)=Y(1,I)
18     DO 10 J=1,KSTEP
19     IF(METHOD.EQ.1) GO TO 4
20
21     C      * FIRST-ORDER ADAMS-BASFORTH METHOD TO START
22     C
23     HA=H/4.D0
24     DO 3 I=1,4
25     CALL LMS(1,HA,YOLD,N,YN,10)

```

```

26      DO 2 L=1,N
27      YOLD(1,L)=YN(L)
28      CONTINUE
29      GO TO 6
30      C      **** FIRST-ORDER GENERALIZED-ADAMS-BASIFORTH METHOD TO START ****
31      C      **** NLMS(I1,H,YOLD,N,YN,IQ,IL,IT,IAT)
32      C      **** CALL NLMS(I1,H,YOLD,N,YN,IQ,IL,IT,IAT)
33      C      **** DO 5 I=1,N
34      C      Y(J+1,I)=YN(I)
35      C      YOLD(I,I)=YN(I)
36      C      CONTINUE
37      C      ALPHA(1)=AL
38      C      RETURN
39      C      END
40
*
* SUBROUTINE DIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,Y,YZERO,IG,IV,METHOD,
* INDEX,IAT)
* **** DIFEQ IS CALLED BY MAIN PROGRAM WHOSE ARGUMENTS ARE ALREADY *
* DEFINED IN THE MAIN PROGRAM WHERE IG=IGFN, IV=IPC
* **** DIFEQ CALLS 4 SUBROUTINES
*      START(KSTEP,...,IAT) --> A STARTER
*      LMS (KSTEP,...,I) --> LINEAR MULTISTEP
*      NLMS (KSTEP,...,IAT) --> NONLINEAR MULTISTEP
*      PRINT(TEA,YNEW) --> FOR USER TO PRINTOUT RESULTS
* **** SOLUTION VECTOR Y(T) IS YNEW(I) OR Y(KSTEP+1,I)
* **** BASED ON USER-SUPPLIED INPUTS, DIFEQ SETS UP THE ITERATIVE
* PROCEDURE, CONTROLS STARTER, STEP-SIZE CHANGES, PREDICTOR-
* CORRECTOR, CORRECTOR'S CONVERGENCE, PRINTOUTS AND CALLS FOR
* THE REQUIRED METHODS
*
1 2
5 4
9
10
11
12
13
14
15
16
17
18
19

```

```

20 COMMON A(20,20),ALPHA(4),T(20)
21 DIMENSION G(20),Y(4,20),YN(20),YOLD(4,20),YZERO(20)
22 DOUBLE PRECISION A,ALPHA,ANORM,ERR,G,H,HMAX,HMIN,T,TEA,TMAX
23 DOUBLE PRECISION TZERO,Y,YN,YNEW,YOLD,YZERO
24 DATA HMIN,IZERO,IONE7,1D-11,0,1/
25 LMT=3
26 WRITE (4,10)
27 10 FORMAT (1H1)
28 DO 40 I=1,N
29 40 Y(I,I)=YZERO(I)
30 ITER=0
31 ISTEP=0
32 TZERO=T(1)
33 49 ITER=ITER+1
34 IH=0
35 IMIN=0
36 50 CONTINUE
37 C **** EVALUATE A(T) AT T=T(0) IF IAT NOT 0
38 C * IF KSTEP GT 1, CALLS START
39 C ****
40 C ****
41 IF (IAT.NE.0) CALL AFNT(A,N,T(1))
42 IF (KSTEP.EQ.1) AND . INDEX.EQ.0) GO TO 60
43 CALL START(KSTEP,H,N,Y,YN,YOLD,IG,METHOD,IAT)
44 TEA=T(1)+DBLE (FLOAT(KSTEP))*H
45 IH=IH+1
46 DO 61 J=1,KSTEP
47 T(J+1)=T(J)+H
48 IMP=KSTEP+1
49 DO 62 J=1,IMP
50 DO 62 I=1,N
51 62 YOLD(J,I)=Y(J,I)
52 C ****
53 C * METHODE=0 -- LMS, METHOD=1 -- NLMS
54 C * FIXED-STEP-SIZE   IV=0, INDEX=0    PREDICTOR
55 C *                   IV=0, INDEX=1    CORRECTOR
56 C * VARIABLE-STEP-SIZE IV NE 0,      PREDICTOR-CORRECTOR

```

```

57      C ****
58      IF(METHOD.0) 60 TO 64
59      IF(INDEX.EQ.0) CALL LMS(KSTEP,H,YOLD,N,YN,IZERO)
60      IF(IV.EQ.0.AND,INDEX.EQ.1) CALL LMS(KSTEP,H,YOLD,N,YN,IONE)
60      GO TO 59
61      64 IF(IV.EQ.0.AND,INDEX.EQ.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IZERO,
62      *          IH,IG,IAT)
63      IF(IV.EQ.0.AND,INDEX.EQ.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IONE,
64      *          IH,IG,IAT)
65      IF(IV.NE.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IZERO,IG,IAT)
66      DO 66 I=1,N
67      YOLD(KSTEP+1,I)=YN(I)
68      IF(IV.NE.0.OR, IAT.NE.0) GO TO 69
69      IF(LMT.EQ.1) GO TU 69
70      DO 68 I=1,N
71      YNEW(I)=YN(I)
72      GO TO 82
73      C ****
74      C   CORRECTOR AT MOST CORRECT 3 TIMES
75      C   STEP-SIZE CHANGED BY A FACTOR OF 2
76      C ****
77      69 ICORR=0
78      70 ICORR=ICORR+1
79      IF(ICORR.LE. LMT) GO TO 75
80      H=H/2.0D0
81      IF(H .LT. HMIN) HMIN
82      IF(H .LT. HMIN) IMIN=IMIN+1
83      IF(IMIN .GT. 3) WRITE(4,1170)
84      1170 FORMAT(3X,39HH REACHED HMIN, NO CONVERGENCE POSSIBLE)
85      IF(IMIN .GT. 3) STOP
86      T(1)=TZERO
87      DO 71 I=1,N
88      Y(I,I)=YZERO(I)
89      GO TO 50
90      75 IF(METHOD.EQ.0) CALL LMS(KSTEP,H,YOLD,N,YN,IONE)
91      IF(METHOD.GT.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IONE,ICURR,IG,IAT)
92      IF(LMT.NE. 1) GO TO 75
93

```

```

94      DO 77 I=1,N
95      YNEW(I)=YN(I)
96      GO TO 82
97      76 CONTINUE
98      ANORM=0.0D0
99
100     C      **** TEST CORRECTOR'S CONVERGENCE, USING MAX NORM
101     C      **** DO 80 J=1,N
102     C      G(J)=(YOLD(KSTEP+1,J)-YNEW(J))/YOLD(KSTEP+1,J)
103     C      IF(ANORM .GT. DABS(G(J))) GO TO 80
104     C      ANORM=DABS(G(J))
105     C      CONTINUE
106     C      IF(ANORM-ERR) 82,82,1182
107     C      IF(ANORM-ERR) 81,81,1182
108     C      YOLD(KSTEP+1,J)=YNEW(J)
109     C      GO TO 70
110     C      82 DO 83 I=1,N
111     C      Y(KSTEP+1,I)=YNEW(I)
112     C      83
113     C      *RESULTS Y(TEA) IN YNEW(I) AND Y(KSTEP+1,I)
114     C      * CALL PRINT FOR USER TO PRINT RESULTS
115     C      * CALL PRINT(H,TEA,YNEW,N,ITER)
116     C      *CONTINUE
117     C      IF(TEA .GT. TMAX) RETURN
118     C      DO 85 J=1,KSTEP
119     C      DO 85 I=1,N
120     C      Y(J,I)=Y(J+1,I)
121     C      IF(J.EQ.1) YZERO(I)=Y(J,I)
122     C      85 CONTINUE
123     C      T(1)=T(2)
124     C      TZERO=T(1)
125
126     C      IF(IV.EQ.0 .AND. INDEX.EQ.1) LMT=1
127     C      IF(IV.EQ.0 .AND. INDEX.EQ.1) IH=0
128     C      IF(ILMT .EQ. 1) IH=0
129     C      IF(IV.EQ.0) GO TO 60
130

```

```

131      T(1)=TEA
132      TZERO=T(1)
133      DO 86 I=1,N
134      YZERO(I)=Y(KSTEP+1,1)
135      86      Y(I,I)=Y(KSTEP+1,I)
136      *****
137      C      * MAINTAIN SUCCESSFUL H CONSTANTLY FOR 3 TIMES BEFORE DOUBLING *
138      C      *****
139      ISTEP=ISTEP+1
140      IF(ISTEP .LT. 3) GO TO 49
141      ISTEP=0
142      H=2.D0*H
143      IF(H .GT. HMAX) H=HMAX
144      GO TO 49
145      END

1      C      SUBROUTINE NLMS(KSTEP,H,Y,N,YN,INDEX,IS,IT,IAT)
2      C      *****
3      C      * NONLINEAR MULTISTEP ALGORITHM(NLMS)
4      C      * CALLED BY DIFEQ OR START
5      C      * ARGUMENTS ALREADY DEFINED IN MAIN PROGRAM
6      C      * THIS PROGRAM CALLS 3 SUBROUTINES
7      C      * INVERT(AH,...,P1)
8      C      * GFN(G,...,NC)
9      C      * PADE(A,...,NC)
10     C      * SOLUTION VECTOR IS STORED IN YN(I)
11     C      *****
12     C      COMMON A(20,20),ALPHA(4),T(20)
13     C      DIMENSION AH(20,20),AH2(20,20),AH3(20,20),AH4(20,20)
14     C      DIMENSION EH(20,20),E2AH(20,20),E3AH(20,20),G(20)
15     C      DIMENSION P(20,20),P1(20,20),PH1(4,20,20),PH1(4,20,20),UNIT(20,20)
16     C      DIMENSION W1(4,20,20),W2(4,20,20),W3(4,20,20),W4(4,20,20)
17     C      DIMENSION AT(20,20),Y(4,20),Y'(20)
18     C      DOUBLE PRECISION A,AH,AH2,AH3,AH4,ALPHA,AT,EAH,E2AH,E3AH,G,H,H2,H3
19     C      DOUBLE PRECISION P,P1,PH1,GHI,T,T1,T2,T3,T4,UNIT,W1,W2,W3,W4,Y,YN

```

TR 5341

```
20
21      C
22      C      * IS IS AN INDICATOR
23      C      * PROGRAM DOES INITIALIZATION WHEN IS IS 1
24      C      * PROGRAM CALCULATES AND SAVES AH, EXP(AH), A INVERSE, PHI FN
25      C      * IS.GT.1 ABOVE CALCULATIONS ARE BYPASSED
26      C
27      IF (IS .GT. 1) GO TO (100,200,300), KSTEP
28      DO 1 I=1,N
29      DO 2 J=1,N
30      P1(I,J)=0.00
31      UNIT(I,J)=0.00
32      EAH(I,J)=0.00
33      E2AH(I,J)=0.00
34      E3AH(I,J)=0.00
35      AH2(I,J)=0.00
36      AH3(I,J)=0.00
37      AH4(I,J)=0.00
38      1 UNIT(I,I)=1.000
39      DO 3 I=1,N
40      DO 3 J=1,N
41      DO 3 K=1,N
42      PHI(I,J,K)=0.00
43      3 AH(I,J,K)=H*A(I,J)
44      DO 6 I=1,N
45      DO 6 J=1,N
46      DO 6 AH(I,J)=H*A(I,J)
47      6 AH(I,J)=H*A(I,J)
48      GO TO (100,200,300), KSTEP
49      100 CONTINUE
50      * NONLINEAR MULTISTEP STARTS HERE.
51      * BEGINNING SECTION DOES INITIALIZATION
52      100 132 I=1,N
53      DO 133 J=1,N
54      P1(I,J)=0.00
55      YN(I)=0.00
56      PHI(2,I,I)=0.00
```

```

57   132 PHI(3,I,1)=0.D0
58   IF (IS.GT.1 .AND. INDEX.EJ.0) GO TO 131
59   IF (IS.GT.1 .AND. INDEX.EQ.1) GO TO 170
60   C **** P1 CONTAINS (AH)**(-1), EAH CONTAINS EXP(AH)
61   C ****
62   C ****
63   IF (N-1) 120,1120,120
64   1120 P1(I,1)=1.000/AH(I,1)
65   GO TO 122
66   120 CALL INVERT(AH,N,P1)
67   122 IF (N-1) 123,1123,123
68   1123 EAH(I,1)=DEXP(A(I,1)*H)
69   GO TO 125
70   123 CALL PAUE(Am,EAH,E2AH,E3AH,G,I)
71   125 DO 103 I=1,N
72   DO 103 J=1,N
73   103 AH(I,J)=ALPHA(I)*EAH(I,J)+UNIT(I,J)
74   C **** EXPLICIT NLM-1-STEP METHODS
75   C **** DO LOOP 105 CALCULATES PHI(I,0)
76   C **** LOOP 108 OR 110 COMPUTES FINAL Y(N+1)
77   C ****
78   C ****
79   C ****
80   IF (INDEX .NE. 0) GO TO 150
81   IF (IT .EQ. 0) GO TO 109
82   DO 105 K=1,N
83   DO 105 I=1,N
84   105 PHI(I,I,J)=PHI(I,I,J)-P1(I,K)*AH(K,J)
85   131 CALL GFN(G0H,N,Y,KSTEP,T,A)
86   LC 108 I=1,N
87   DO 106 J=1,N
88   106 Y(I,J)=YIN(I)-ALPHA(I)*EAH(I,J)*Y(I,J)+H*PHI(I,I,J)*G(J)
89   RETURN
90   109 DO 110 I=1,N
91   DO 110 J=1,N
92   110 Y(I,J)=YIN(I)-ALPHA(I)*EAH(I,J)*Y(I,J)
93   RETURN

```

```

94      C      **** IMPLICIT NLM-1-STEP METHODS
95      C      * COMPUTE PHI(I,0), PHI(1,1)
96      C      * FINAL RESULTS ARE CALCULATED IN LOOP 166 OR 168
97      C      ****
98      C      150 IF(N=1) 152,1152,152
99      1152 AH2(1,1)=P1(1,1)*P1(1,1)
100     GO TO 153
101     DO 154 I=1,N
102     DO 154 J=1,N
103     DO 154 K=1,N
104     AH2(I,J)=AH2(I,J)+P1(1,K)*P1(K,J)
105     154 IF(IT .EQ. 0) GO TO 167
106     DO 160 I=1,N
107     DO 161 J=1,N
108     DO 162 K=1,N
109     PHI(1,I,J)=PHI(1,I,J)+ALPHA(1)*H*A(I,K)*EAH(K,J)
110
111     PHI(1,I,J)=PHI(1,I,J)-AH(I,J)
112     E2/H(I,J)=AH(I,J)+H*A(I,J)
113     160 CONTINUE
114     170 K2=KSTEP+1
115     IF(IT .EQ. 0) GO TO 167
116     DO 163 I=1,N
117     DO 164 K=1,K2
118     CALL GFN(G,H,N,Y,K,T,A)
119     DO 165 J=1,N
120     IF(K.EQ.1) PHI(3,I,1)=PHI(3,I,1)+PHI(1,I,J)*G(J)
121     IF(K.EQ.2) PHI(2,I,1)=PHI(2,I,1)+E2AH(I,J)*G(J)
122     165 PHI(3,I,1)=PHI(3,I,1)+PHI(2,I,1)
123     164 CONTINUE
124     163
125     DO 166 I=1,N
126     YN(I)=YN(I)-H*AH2(I,K)*PHI(3,K,1)-ALPHA(1)*EAH(I,K)*Y(I,K)
127     GO TO 172
128     DO 166 I=1,N
129     YN(I)=YN(I)-ALPHA(1)*EAH(I,K)*Y(I,K)
130

```

```

131   C **** IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION, *
132   C * EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1)
133   C ****
134   C 172 IF(IAT .EQ. 0) RETURN
135   C 176 THET(1)+H
136   C CALL AFNT(AT,N,Th)
137   DO 173 I=1,N
138   G(I)=0.00
139   UO 173 J=1,N
140   173 G(I)=G(I)+(AT(I,J)-A(I,J))*Y(2,J)
141   DO 174 J=1,N
142   UO 174 I=1,N
143   GHI(4,4,I)=0.00
144   174 GHI(4,4,I)=GHI(4,4,I)+E2AH(I,J)*G(J)
145   175 YN(I)=YN(I)-H*AH2(I,J)*GHI(4,4,J)
146   UO 175 I=1,N
147   UO 175 J=1,N
148   175 YN(I)=YN(I)-H*AH2(I,J)*GHI(4,4,J)
149   RETURN
150   200 CONTINUE
151   C ****
152   C * NONLINEAR MULTI-2-STEP STARTS HERE
153   C * BEGINNING SECTION DOES INITIALIZATION
154   C ****
155   UO 240 I=1,N
156   DO 241 J=1,N
157   P1(I,J)=0.00
158   241 YN(I)=0.00
159   240 PHI(4,1,I)=0.00
160   IF (IS .GT. 1) GO TO 212
161   C ****
162   C * LHN CONTAINS EXP(AH). E2AH CONTAINS EXP(2AH)
163   C ****
164   IF (H=1) 208,220B,208
165   208 LAH(1,I)=EXP(AH(1,I))
166   L2AH(1,I)=EAN(1,I)*EAN(1,I)

```

```

167      GO TO 212
168      208 CALL PADE(A,H,EAH,E2AH,E3AH,G,N)
169      H2=H+H
170      CALL PAUL(A,H2,E2AH,E3AH,AH4,6,N)
171      212 IF (INDEX .GT. 0) GO TO 260
172      ****
173      C   * EXPLICIT ILM-2-STEP METHOD
174      C   * AH2 CONTAINS (AH)**(-2)
175      C   ****
176      C   IF (IS .GT. 1) GO TO 234
177      251 IF (N-1) 201,2201,201
178      2201 AH(1,1)=1.0D0/AH(1,1)**2
179      GO TO 206
180      201 CALL INVERT(AH,N,P1)
181      DO 203 I=1,N
182      UO 203 JE1,N
183      UO 203 KE1,N
184      164 AH2(I,J)=AH2(I,J)+P1(I,K)*P1(K,J)
185      165 IF (IT .EQ. 0) GO TO 235
186      ****
187      C   * AT THE FINISH OF LOOP 215 --
188      C   * PHI(1,I,J) CONTAINS PHI(1,0), PHI(2,I,J) CONTAINS PHI(2,0)
189      C   * FINAL Y(N+1) ARE CALCULATED IN LOOP 232
190      C   ****
191      206 UO 213 I=1,N
192      UO 213 JE1,N
193      213 P1(I,J)=ALPHA(1)*E2AH(I,J)+ALPHA(2)*EAH(I,J)+UNIT(I,J)
194      DO 215 I=1,N
195      UO 215 JE1,N
196      UO 215 K=1,N
197      PHI(1,I,J)=PHI(1,I,J)+ALPHA(1)*AH(I,K)*E2AH(K,J)
198      215 PHI(2,I,J)=PHI(2,I,J)+ALPHA(2)*AH(I,K)*EAH(K,J)
199      UO 216 I=1,N
200      UO 219 J=1,N
201      PHI(1,I,J)=PHI(1,I,J)-P1(I,J)-AH(I,J)
202      PHI(2,I,J)=PHI(2,I,J)+P1(I,J)+2.000*AH(I,J)

```

```

203      P1(1,1)=0.D0
204      UU 220 K=1,KSTEP
205      CALL UFN(G,H,I,J,K,T,A)
206      DO 221 I=1,N
207      DO 221 J=1,N
208      YN(I)=YN(I)+PHI(K,I,J)*G(J)*H
209      CONTINUE
210      DO 230 I=1,N
211      DO 230 J=1,N
212      P1(1,1)=P1(1,1)-AH2(I,J)*YN(J)
213      DO 232 I=1,N
214      IF(I.EQ.0) P1(1,1)=0.00
215      DO 233 J=1,N
216      PHI(4,1,I)=PHI(4,1,I)-ALPHA(2)*EAH(I,J)*Y(2,J)-ALPHA(1)*E2AH(I,J)*
217      Y(1,J)
218      YN(I)=P1(1,I)+PHI(4,1,I)
219      IF(IAT .EQ. 0) RETURN
220      DO 290 I=1,N
221      DO 290 J=1,N
222      E2AH(I,J)=PHI(2,I,J)
223      GO TO 170
224      C *****
225      C * IMPLICIT NONLINEAR MULTI-2-STEP METHODS
226      C * NEXT BELOW 267, AH3 CONTAINS (AH)**(-3)
227      C *****
228      CONTINUE
229      IF(IIS .GT. 1) GO TO 284
230      IF(N-1) 262,2262,262
231      AH2(1,1)=AH(1,1)*AI,(1,1)
232      AH3(1,1)=1.0D0/(AH2(1,1)*AH(1,1))
233      GO TO 263
234      DO 264 I=1,N
235      DO 264 J=1,N
236      DO 264 K=1,N
237      AH2(1,J)=AH2(1,J)+(AH(1,K)*AH(K,J))
238      GO 237 I=1,N

```

TR 5341

```
239 DO 267 J=1,N
240 AH4(I,J)=0.D0
241 U0 267 K=1,N
242 267 AH4(I,J)=AH4(I,J)+AH2(I,K)*AH(K,J)
243 CALL INVERT(AH4,N,AH3)
244 CONTINUE
245 263 IF(IIS .GT. 1) GO TO 279
246 IF(II .EQ. 0) GO TO 285
247 C
248 C      END OF LOOP 275. PHI(L,I,J) CONTAINS PHI(2,L), L=0,1,2
249 C      FINAL Y(N+1) ARE CALCULATED IN LOOP 282 OR 286
250 C
251 DO 270 I=1,N
252 U0 270 J=1,N
253 W1(1,I,J)=UNIT(I,J)-1.5D0*AH(I,J)+AH2(I,J)
254 W1(2,I,J)=UNIT(I,J)-0.5D0*AH(I,J)
255 W1(3,I,J)=UNIT(I,J)+0.5D0*AH(I,J)
256 W2(1,I,J)=-2.D0*(UNIT(I,J)-AH(I,J))
257 W2(2,I,J)=-2.D0*(UNIT(I,J)+AH2(I,J))
258 W2(3,I,J)=-2.D0*(UNIT(I,J)+AH(I,J))
259 W3(1,I,J)=W1(2,I,J)
260 W3(2,I,J)=W1(3,I,J)
261 W3(3,I,J)=UNIT(I,J)+1.5D0*AH(I,J)+AH2(I,J)
262 U0 272 I=1,N
263 DO 273 J=1,N
264 DO 274 L=1,N
265 GHI(1,I,J)=GHI(1,I,J)+ALPHA(1)*W1(1,I,L)*E2AH(L,J)
266 +ALPHA(2)*W1(2,I,L)*EAH(L,J)
267 GHI(2,I,J)=GHI(2,I,J)+ALPHA(1)*W2(1,I,L)*E2AH(L,J)
268 +ALPHA(2)*W2(2,I,L)*EAH(L,J)
269 GHI(3,I,J)=GHI(3,I,J)+ALPHA(1)*W3(1,I,L)*E2AH(L,J)
270 +ALPHA(2)*W3(2,I,L)*EAH(L,J)
271 CONTINUE
272 GHI(1,I,J)=GHI(1,I,J)+W1(3,I,J)
273 GHI(2,I,J)=GHI(2,I,J)+W2(3,I,J)
274 GHI(3,I,J)=GHI(3,I,J)+W3(3,I,J)
```

```

      C75
      C76      CONTINUE
      C77      UJ 275 L=1,N
      C78      DC 275 I=1,N
      C79      UJ 275 JE1,N
      C80      UJ 275 KE1,N
      C81      PHI(L,I,J)=H1(L,I,J)-AH3(I,K)*UH1(L,K,J)
      C82      UJ 280 I=1,N
      C83      DC 280 K=1,J
      C84      CALL GFN(G,H,N,Y,K,T,A)
      C85      DO 282 J=1,N
      C86      YN(I)=YN(I)+PHI(K,I,J)*G(J)*R
      C87      IF (K .EQ. 3) YN(1)=Y1(I)-ALPHA(1)*E2AH(I,J)*Y(1,J)-ALPHA(2)*EAH(I,
      C88      SJ)*Y(2,J)
      C89      CONTINUE
      C90      GU TO 293
      C85      UJ 286 I=1,N
      C91      DO 286 JE1,N
      C92      YN(I)=YN(I)-ALPHA(1)*E2AH(I,J)*Y(1,J)-ALPHA(2)*EAH(I,J)*Y(2,J)
      C93      IF (IAT .EQ. 0) RETURN
      C94      *****
      C95      * IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION
      C96      * EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1)
      C97      *****
      C98      *****
      C97      T1=T(1)+H
      C99      T2=T1+H
      C00      CALL AFNT(P,...,T1)
      C01      CALL AFNT(AT,N,T2)
      C02      DO 294 I=1,N
      C03      G(I)=0.0D0
      C04      P1(I,1)=0.0D0
      C05      DO 294 J=1,N
      C06      G(I)=G(I)+(P(I,J)-A(J,I))*Y(2,J)
      C07      P1(I,1)=P1(I,1)+(AT(I,J)-A(I,J))*Y(3,J)
      C08      DO 295 I=1,N
      C09      P1(I,1)=0.0D0
      C10

```

TR 5341

```
311 DO 295 J=1,N
312 P1(2,1)=P1(2,1)+PH1(2,I,J)*G(J)+PH1(3,I,J)*P1(1,J)
313 IF(KSTEP.EQ.3) GO TO 298
314 DO 296 I=1,N
315 YN(I)=YN(I)+H*P1(2,I)
316 RETURN
317 298 DO 299 I=1,N
318 YN(I)=YN(I)-H*P1(2,I)
319 RETURN
320 300 CONTINUE
321 C **** NONLINEAR MULTI-3-STEP STARTS HERE
322 C * BEGINNING SECTION DOES INITIALIZATION
323 C * BEFORE 303, THE FOLLOWING RESULTS ARE STORED.**
324 C * EAH--EXP(AH), E2AH--EXP(2AH), E3AH--EXP(3AH), AH3--(AH)**(-3)
325 C ****
326 C ****
327 KUP=KSTEP
328 IF(INDEX.EQ.1) KUP=KSTEP+1
329 UO 320 I=1,N
330 YN(I)=0.D0
331 IF(1S.GT.1) GO TO 321
332 IF(N-1) 302,3302,302
333 EAH(1,1)=UEXP(A(1,1)*H)
334 E2AH(1,1)=EAH(1,1)*EAH(1,1)
335 E3AH(1,1)=E2AH(1,1)*EAH(1,1)
336 AH2(1,1)=AH(1,1)*AH(1,1)
337 IF(INDEX.EQ.1) GO TO 350
338 AH3(1,1)=1.0D0/(AH2(1,1)*AH(1,1))
339 GO TO 303
340 302 DO 330 I=1,N
341 DO 330 J=1,N
342 DO 330 K=1,N
343 AH2(I,J)=AH2(I,J)+AH(I,K)*AH(K,J)
344 DO 333 I=1,N
345 DO 333 J=1,N
346 AH4(I,J)=0.WC
```

```

347 UC 333 K=1,N
348 AH4(I,J)=AH4(I,J)+AH2(I,K)*AH(K,J)
349 IF(INU<X .EQ. 1) GO 10 351
350 CALL INVERT(AH4)N,AH3)
351 CALL PADE(A,H,EAH,E2AH,E3AH,G,N)
352 H2=H+H
353 CALL PADE(A,H2,E2AH,E3AH,AH4,G,N)
354 H3=H2+H
355 CALL PADE(A,H3,E3AH,AH4,W1(1,1,1),G,N)
356 303 IF(IT .EQ. 0) GO TO 370
357 C ****
358 C * CALCULATE PHI(3,K), K=0,1,2. RESULTS IN PHI(K,I,J)
359 C ****
360 DO 304 I=1,N
361 DO 304 J=1,N
362 W1(1,I,J)=UNIT(I,J)-1.5D0*AH(I,J)+AH2(I,J)
363 W1(2,I,J)=UNIT(I,J)-0.5D0*AH(I,J)
364 W1(3,I,J)=UNIT(I,J)+0.5D0*AH(I,J)
365 W1(4,I,J)=UNIT(I,J)+1.5D0*AH(I,J)+AH2(I,J)
366 W2(1,I,J)=-2.0D0*(UNIT(I,J)-AH(I,J))
367 W2(2,I,J)=-2.0D0*UNIT(I,J)+AH2(I,J)
368 W2(3,I,J)=-2.0D0*(UNIT(I,J)+AH(I,J))
369 W2(4,I,J)=-2.0D0*UNIT(I,J)-4.0D0*AH(I,J)-3.D0*AH2(I,J)
370 W3(1,I,J)=UNIT(I,J)-0.5D0*AH(I,J)
371 W3(2,I,J)=UNIT(I,J)+0.5D0*AH(I,J)
372 W3(3,I,J)=W1(4,I,J)
373 W3(4,I,J)=UNIT(I,J)+2.5D0*AH(I,J)+3.D0*AH2(I,J)
374 304
375 DO 307 I=1,N
376 DO 308 J=1,N
377 DO 309 K=1,N
378 QHI(1,I,J)=QHI(1,I,J)+ALPHA(1)*W1(1,I,K)*E3AH(K,J)
379 +ALPHA(2)*W1(2,I,K)*E2AH(K,J)
380 +ALPHA(3)*W1(3,I,K)*EAH(K,J)
381 +ALPHA(1)*W2(1,I,K)*E3AH(K,J)
382 +ALPHA(2)*W2(2,I,K)*E2AH(K,J)
383 +ALPHA(3)*W2(3,I,K)*EAH(K,J)
384 QHI(2,I,J)=QHI(2,I,J)+ALPHA(1)*W3(1,I,K)*E3AH(K,J)
385 +ALPHA(2)*W3(2,I,K)*E2AH(K,J)
386 +ALPHA(3)*W3(3,I,K)*EAH(K,J)

```

```

386      IF (KUP .NE. 4) GO TO 309
387      QHI(4,I,J)=QHI(4,I,J)+ALPHA(1)*W4(1,I,K)*E3AH(K,J)
388      *          +ALPHA(2)*W4(2,I,K)*E2AH(K,J)
389      *          +ALPHA(3)*W4(3,I,K)*EAH(K,J)
390      309      CONTINUE
391      QHI(1,I,J)=QHI(1,I,J)+W1(4,I,J)
392      QHI(2,I,J)=QHI(2,I,J)+W2(4,I,J)
393      QHI(3,I,J)=QHI(3,I,J)+W3(4,I,J)
394      IF (KUP.EQ.4) QHI(4,I,J)=QHI(4,I,J)+W4(4,I,J)
395      308      CONTINUE
396      307      CONTINUE
397      DO 310 K=1,KUP
398      DO 310 I=1,N
399      DO 310 J=1,N
400      UU 310 L=1,N
401      IF (INDEX.EQ.0) PHI(K,I,J)=PHI(K,I,J)-AH3(I,L)*QHI(K,L,J)
402      IF (INDEX.EQ.1) PHI(K,I,J)=PHI(K,I,J)-AH4(I,L)*QHI(K,L,J)
403      UU 314 K=1,KUP
404      CALL GFN(G,H,N,Y,K,T,A)
405      DO 315 I=1,N
406      DO 316 J=1,N
407      C
408      C          *****
409      C          * CALCULATE FINAL Y(N+1)
410      C          *****
411      YN(I)=YN(I)+H*PHI(K,I,J)*G(J)
412      316 IF (K .EQ. KUP) YN(I)=YN(I)-ALPHA(3)*EAH(I,J)*Y(3,J)-ALPHA(2)*E2AH(
413      S,I,J)*Y(2,J)-ALPHA(1)*E3AH(I,J)*Y(1,J)
414      315 CONTINUE
414      314 CONTINUE
415      GO TO 340
416      370 UU 371 I=1,N
417      DO 371 J=1,N
418      371 YN(I)=YN(I)-ALPHA(3)*EAH(I,J)*Y(3,J)-ALPHA(2)*E2AH(I,J)*Y(2,J)-ALFA
419      1MA(1)*E3AH(I,J)*Y(1,J)
420      340 IF (IAT .EQ. 0) RETURN
421      IF (INDEX .EQ. 0) GO TO 297

```

```

422      C **** TO SEE IS A A FUNCTION OF T ****
423      C **** **** **** **** **** **** **** ****
424      C **** **** **** **** **** **** **** ****
425      C **** T1=T(1)+H
426      C **** T2=T1+H
427      C **** T3=T2+H
428      C CALL AFNT(AT,N,Y1)
429      C CALL AFNT(P,N,T2)
430      C CALL AFNT(P1,N,T3)
431      DO 341 I=1,N
432      G(I)=0.D0
433      GH1(4,3,1)=0.D0
434      GH1(4,4,1)=0.D0
435      DO 341 J=1,I
436      G(I)=G(I)+(AT(I,J)-A(I,J))*Y(J)
437      GH1(4,3,1)=GH1(4,3,I)+(P(I,J)-A(I,J))*Y(3,J)
438      GH1(4,4,1)=GH1(4,4,I)+(P1(I,J)-A(I,J))*Y(4,J)
439      DO 342 I=1,N
440      DO 342 J=1,N
441      YN(I)=YN(I)+H*(PHI(2,I,J)*G(J)+PHI(3,I,J)*QHI(4,I,J)+PHI(4,I,J)*O1
442      S1(4,4,J)
443      RETURN
444      C **** **** **** **** **** **** **** **** ****
445      C **** CALCULATE IMPLICIT PHI FUNCTION, PHI(3,J), J=0,1,2,3
446      C **** **** **** **** **** **** **** **** ****
447      C 350 AH3(1,1)=AH2(1,1)*AH(1,1)
448      AH4(1,1)=1.000/(AH3(1,1)*AH(1,1))
449      GO TO 357
450      DO 354 I=1,N
451      DO 354 J=1,N
452      EAH(I,J)=0.D0
453      AH3(I,J)=AH4(I,J)
454      DO 354 K=1,N
455      EAH(I,J)=EAH(I,J)+AH4(I,K)*AH(K,J)
456      CALL INVERT(EAH(1,1),AH4)
457      CALL PADE(A,H,E2AH,E3AH,G01)
458      H2=H+H
459      CALL PADE(A,H2,E2AH,E3AH,G11)
460      H3=H2+H

```

```

461      CALL PADE(A,H3,E3AH,W1(1,1,1),W2(1,1,1),G,N)
462      IF(IT.EQ.0) GO TO 370
463      DO 358 I=1,N
464      DO 358 J=1,N
465      W1(1,1,J)=-UNIT(I,J)+2.D0*AH(I,J)-11.D0*AH2(I,J)/6.D0+AH3(I,J)
466      W1(2,1,J)=-UNIT(I,J)+AH(I,J)-AH2(I,J)/3.D0
467      W1(3,1,J)=-UNIT(I,J)+AH2(I,J)/6.D0
468      W1(4,1,J)=-UNIT(I,J)-AH(I,J)-AH2(I,J)/3.D0
469      W2(1,1,J)=3.U0*UNIT(I,J)-5.D0*AH(I,J)+3.D0*AH2(I,J)
470      W2(2,1,J)=3.D0*UNIT(I,J)-2.D0*AH(I,J)-0.5D0*AH2(I,J)+AH3(I,J)
471      W2(3,1,J)=3.D0*UNIT(I,J)+AH(I,J)-AH2(I,J)
472      W2(4,1,J)=3.D0*UNIT(I,J)+4.D0*AH(I,J)+1.5D0*AH2(I,J)
473      W3(1,1,J)=-3.D0*UNIT(I,J)+4.D0*AH(I,J)-1.5D0*AH2(I,J)
474      W3(2,1,J)=-3.DC*UNIT(I,J)+AH(I,J)+AH2(I,J)
475      W3(3,1,J)=-3.D0*UNIT(I,J)-2.D0*AH(I,J)+0.5D0*AH2(I,J)+AH3(I,J)
476      W3(4,1,J)=-3.D0*UNIT(I,J)-5.D0*AH(I,J)-3.D0*AH2(I,J)
477      W4(1,1,J)=-W1(2,1,J)
478      W4(2,1,J)=-W1(3,1,J)
479      W4(3,1,J)=-W1(4,1,J)
480      W4(4,1,J)=UNIT(I,J)+2.D0*AH(I,J)+11.D0*AH2(I,J)/6.D0+AH3(I,J)
481      GO TO 360
482      END

```

```

1      SUBROUTINE LMS(KSTEP,H,Y,N,YH,INDEX)
2      *****
3      **** LINEAR MULTISTEP METHODS(STEP NUMBER.LE.3)
4      **** BETA COEFFICIENTS ARE FORMULATED TO DEPEND UPON THE
5      **** CHARACTERISTIC COEFFICIENTS. ALPHA
6      **** 100 THROUGH 400 CALCULATES BETA. CALCULATIONS ARE NEEDED
7      **** ONLY ONCE
8      **** BETAS OF EXPLICIT METHODS ARE STORED IN B1.
9      **** OF IMPLICIT METHODS, IN B2
10     **** THIS SUBROUTINE IS CALLED BY START OR DIFEQ
11

```

```

12
13      COMMON A(<0,20),ALPHA(4),T(20)
14      DIMENSION Y(4,20),YN(20),B(4),B1(3),B2(4),FN(20)
15      DOUBLE PRECISION A,ALPHA,B,B1,B2,FN,H,T,Y,YN
16      DATA IBETA/0/
17      KSTEP
18      DO 1 I=1,N
19      1   Y(I)=0.00
20      IF(IBETA .GT. 0) GO TO 410
21      GO TO (100,200,300),KSTEP
22      100  B1(1)=1.00
23      B2(1)=0.500
24      B2(2)=B2(1)
25      GO TO 400
26      B1(1)=0.500*ALPHA(2)
27      B1(2)=B1(1)+2.000
28      B2(1)=5.000*ALPHA(2)/12.000+1.00/3.000
29      B2(2)=2.000*ALPHA(2)/3.000+4.000/3.000
30      B2(3)=-ALPHA(2)/12.000+1.000/3.000
31      GO TO 400
32      B1(1)=5.000*ALPHA(2)/12.000+ALPHA(3)/3.000+0.75D0
33      B1(2)=2.000*ALPHA(2)/3.000+4.000*ALPHA(3)/3.000
34      B1(3)=-ALPHA(2)/12.000+ALPHA(3)/3.000+2.25D0
35      B2(1)=3.000*ALPHA(2)/8.000+ALPHA(3)/3.000+3.000/8.000
36      B2(2)=19.000*ALPHA(2)/24.000+4.000*ALPHA(3)/3.000+9.000/8.000
37      B2(3)=-5.000*ALPHA(2)/24.000+ALPHA(3)/3.000+9.000/8.000
38      B2(4)=ALPHA(2)/24.000+3.000/8.000
39      IBETA=1
40      IF(KSTEP .EQ. 1) IBETA=0
41      IF(INLEX .GT. 0) GO TO 500
42      DO 415 I=1,3
43      B(I)=B1(I)
44      CALL FFN(Y,I,0,FR,1)
45      DO 440 J=1,I,
46      YN(J)=YN(J)+H*B(I)*FI(J)
47      CONTINUE

```

```

48      CONTINUE
49      IF(K .EQ. KSTEP) GO TO 465
50      CALL FFN(Y,N,FN,K)
51      DO 460 J=1,N
52      460  YN(J)=YN(J)+H*B(K)*FN(J)
53      465  DO 470 I=1,N
54      470  DO 470 J=1,KSTEP
55      YN(I)=YN(I)-ALPHA(J)*Y(J,I)
56      470  CONTINUE
57      RETURN
58      500 K=KSTEP+1
59      DO 510 I=1,4
60      510 B(I)=B2(I)
61      GO TO 420
62      END

```

```

SUBROUTINE PADE(A,H,P,B,C,COL,N)
*** **** * **** * **** * **** * **** * **** * **** * **** * **** * **** *
C   * CALCULATES MATRIX EXPONENTIAL BY PADE APPROXIMATION
C   * CALLED BY NLMS SUBROUTINE
C   *** **** * **** * **** * **** * **** * **** * **** * **** * **** *
C   DOUBLE PRECISION A(20,20),P(20,20),B(20,20),C(20,20),COL(20)
C   DOUBLE PRECISION BETA,H,HAVE,XNORM
C   DATA BETA/.30600/
C   HAVE=H
DO 2 I=1,N
      DO 1 J=1,N
        B(I,J)=0.0D0
        C(I,J)=0.0D0
        P(I,J)=0.0D0
        CCNTINUE
1     COL(I)=0.0D0
2     CONTINUE
      DO 17 I=1,N
        DO 16 J=1,N
          COL(I)=COL(I)+DABS(A(J,I))
16
17
18
19
20

```

```

21   16 CONTINUE
22   17 CONTINUE
23   XNORM=COL(1)
24   DO 16 I=1,N
25   IF (XNORM .GT. COL(I)) GO TO 18
26   XNORM=COL(I)
27   CONTINUE
28   C      **** COLUMNS NORM IS USED TO SEE WHETHER EXP(A) NEEDS REDUCTION
29   C      * COLUMN NORM IS USED TO SEE WHETHER EXP(A) NEEDS REDUCTION
30   C      ****
31   M=0
32   30 IF (XNORM*H - .1U0) J,20,2C
33   ****
34   C      EXP (A)=(I-(.5+BETA)*A+BETA*A*.2)*(-1)*(I+(.5-BETA)*A)
35   ****
36   3 UO 6 I=1,N
37   DC 5 J=1,N
38   DO 4 K=1,N
39   P(I,J)=P(I,J)+A(I,K)*A(K,J)
40   CONTINUE
41   C(I,J)=(BETA*A(P(I,J)*H-(.5D0+BETA)*A(I,J))*H
42   CONTINUE
43   C(I,I)=C(I,I)+1.0D0
44   CONTINUE
45   CALL INVERT(C,N,B)
46   DO 9 I=1,N
47   DC 10 J=1,N
48   C(I,J)=(.5D0-BETA)*A(I,J)*H
49   P(I,J)=0.0D0
50   CONTINUE
51   C(I,I)=C(I,I)+1.0D0
52   9 CONTINUE
53   DO 12 I=1,N
54   DC 13 J=1,N
55   DO 14 K=1,N
56   P(I,J)=P(I,J)+B(I,K)*C(K,J)
57   14 CONTINUE
58   13 CONTINUE

```

TR 5341

```
59      12 CONTINUE
60      IF (M .EQ. 0) GO TO 40
61      C
62      C
63      C
64      C
65      DO 24 I=1,N
66      DO 25 J=1,M
67      B(I,J)=0.00
68      CONTINUE
69      DO 36 K=1,M
70      DO 27 I=1,N
71      DO 26 J=1,N
72      DO 29 L=1,N
73      B(I,J)=B(I,J)+P(I,L)*P(L,J)
74      CONTINUE
75      CONTINUE
76      CONTINUE
77      DO 31 I=1,N
78      DO 32 J=1,N
79      P(I,J)=B(I,J)
80      B(I,J)=0.00
81      CONTINUE
82      31 CONTINUE
83      36 CONTINUE
84      H=HAVE
85      RETURN
86      H=M/2.00
87      M=M+1
88      DO 54 I=1,N
89      DO 55 J=1,N
90      P(I,J)=0.00
91      55 CONTINUE
92      54 CONTINUE
93      GO TO 30
94      H=HAVE
95      RETURN
96      END
```

```

SUBROUTINE GFN(Ω, H, N, Y, J, T, A)
* * * * * DOUBLE PRECISION A(20,20), Y(4,20), G(20), T(20), H
* * * * * CALCULATES THE G(T,Y). ALL CALCULATIONS ENTER HERE
* * * * * CALLED BY NLMS
* * * * * E.G.   0Y+UT=-100Y+(1+T**2)
* * * * * DEFINE G(1)=1.+T(J)*T(J)
* * * * * RETURN, IJ
* * * * * END

```

```

SUBROUTINE FFN(Y,N,FN,I)
COMMON A(20,20),ALPHA(4),T(20)
DIMENSION FN(20),Y(4,20)
DOUBLE PRECISION A,ALPHA,FN,T,Y
***** CALCULATES THE F(T,Y). ALL CALCULATIONS ENTER HERE
***** CALLED BY SUBROUTINE LMS
* E.G.   UY/DT=-100Y+(1+T**2)
* DEFINE FN(I)=-100.*Y(I,1)+(1.+T(I))*T(I)
***** RETURN
E.I.U

```

```

1      SUBROUTINE AFNT(A,N,T)
2      DOUBLE PRECISION A(20,20),T
3      *****
4      * MATRIX A IS A FUNCTION OF T. ENTER ALL COMPUTATIONS HERE
5      * E.G. A(T)=-100T
6      * DEFINE A(1,1)=-100.*T
7      *
8      RETURN
9      END

```

```

1 SUBROUTINE PRINT(H,T,Y,N,I)
2   DOUBLE PRECISION H,T,Y(20)
3
4   C   **** USENEW FOR USER TO PRINTOUT INTERMEDIATE RESULTS, Y(T) ****
5   C   * USER CAN DEFINE HIS OWN PRINTOUT FORMATS
6   C   **** **** **** **** **** **** **** **** **** **** **** **** ****
7   12 FORMAT(1X,2(E15.6,2X),4X,5(E15.8,2X))
8   1184 WRITE(4,12) H,T,(Y(J),J=1,N)
9
10  84 CONTINUE
11
12  RETURN
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
21
```

INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
ONR, Code 427, 480, 481, 483, 485	5
CNO, Code OP-23T	1
NRL	1
NAVELECSYSCOMQY, Code 03, 051	2
NAVSEASYSCOMHQ, SEA-09G3	4
NAVAIRDEVVCEN	1
NAWPNSCEN	1
DTNSRDC	1
NAVCOASTSYSLAB	1
CIVENGRLAB	1
NELC	1
NUC	1
NAVSEC, SEC-6034	1
NAVPGSCOL	1
APL/UW, Seattle	1
ARL/Penn State	1
DDC, Alexandria	12
Polytechnic Institute of New York, Professor Stanley Preiser	24
Westinghouse Electric Corp., Mr. J. T. Malone	4
Consolidated Edison Company of New York, Inc., Mr. P. Hsiang	1
Mr. R. Casal, Brooklyn, New York	1
RDP Inc., Mr. C. A. Steele, Jr.,	1
Sperry Systems Management, Mr. James Leong	1
Babcock & Wilcox, Mrs. Anne M. Schwartz	1
University of Illinois, Urbana, Dr. S. W. Joshi	1
Clarion State College, Dr. Stephen Gendler	1